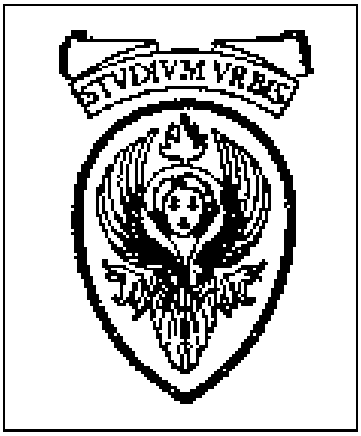


Università di Roma "La Sapienza"
Dispense del Corso di
INTELLIGENZA ARTIFICIALE
Anno Accademico 2000-2001

Luigia Carlucci Aiello
Fiora Pirri

Marzo 2001

Versione 4-riveduta

Università di Roma "La Sapienza"
Dipartimento di Informatica e Sistemistica
via Salaria 113
00186 Roma Italia

Marzo 2001
Versione 4-riveduta

Ai nostri figli: Marco, Luce, Benjamin e Giordano

Prefazione

Ci sono molti modi di esporre la logica, a seconda che si pensi a lettori filosofi, matematici, informatici o ricercatori di intelligenza artificiale[1]. Non che la logica sia diversa (anche se spesso si sente parlare di logica matematica, logica filosofica, eccetera), ma piuttosto diverso è l'uso che della logica si intende fare nei diversi campi.

In matematica la logica è strumento per formalizzare il ragionamento; di qui la sua forza e il suo limite per quanto riguarda l'applicabilità all'informatica. Infatti il ragionamento matematico fa riferimento a realtà idealizzate, dove le verità sono note con certezza, eterne e immutabili, dove il sillogismo è un meccanismo adeguato per trarre conseguenze.

In informatica la logica ha molteplici usi: dalla verifica di programmi[2], alla dimostrazione automatica di teoremi, alla rappresentazione della conoscenza in intelligenza artificiale e alla rappresentazione del ragionamento quotidiano.

La differenza che emerge immediatamente rispetto alla matematica è la natura costruttiva dei metodi risolutivi: in informatica interessano metodi costruttivi, e possibilmente efficienti. In intelligenza artificiale, non è possibile ragionare in una situazione di "mondo chiuso", cioè a conoscenza completa e immutabile nel tempo: se si vuole riprodurre il ragionamento umano bisogna individuare calcoli che permettano inferenze anche in situazione di conoscenza incompleta, mutevole, e che sappiano ragionare di azioni, cambiamenti e di verità che variano nel tempo.

La ricerca in logica e deduzione automatica è stata molto attiva in questo secolo e ha portato allo sviluppo di sistemi di deduzione e alla realizzazione di sistemi di

[1] Talvolta per "intelligenza artificiale" useremo l'acronimo AI, dall'inglese *Artificial Intelligence*.

[2] Senza dimenticare il ruolo della logica proposizionale nella teoria dei circuiti logici.

verifica automatica (o interattiva) di proprietà[3].

Dall'altra parte, la ricerca su semantica dei linguaggi e sistemi di programmazione da un lato, di intelligenza artificiale dall'altro, hanno fatto emergere dei problemi nuovi, non trattati da quella che è diventata universalmente nota come *logica classica*. Tra questi i più significativi sono: il fatto che il tempo passa e si deve poter parlare di proprietà che iniziano a valere o cessano di valere, il fatto che conoscenza si aggiunge attraverso l'osservazione e l'esperienza e a ogni istante si deve essere in grado di ragionare al meglio di ciò che si conosce.

Per molti anni i ricercatori in intelligenza artificiale hanno rifiutato la logica come formalismo di rappresentazione della conoscenza, proprio perché ne vedevano i limiti derivanti dalle sue origini di strumento per ragionare di oggetti matematici, quindi idealizzati. La ricerca si è perciò rivolta alla definizione di formalismi di rappresentazione alternativi, che permettessero di superare i limiti della logica. Anche se questa ricerca ha dato dei frutti interessanti, lo scotto che è stato pagato, per anni, è stato quello di avere a che fare con formalismi dalla sintassi incerta e dalla semantica oscura. Si è tornati quindi con rinnovato interesse alla logica – dagli anni '80 soprattutto – con due approcci: da una parte i fautori della logica del primo ordine che hanno sostenuto e sostengono che tutto si può – quindi si deve – rappresentare in logica del primo ordine, eventualmente con varie estensioni, dall'altra i fautori della messa a punto di nuove logiche ad hoc per risolvere quei problemi che nella logica classica non trovano facile espressione. Sono state quindi sviluppate varie forme di logiche modali per tener conto delle modalità che più frequentemente si incontrano in intelligenza artificiale, quali la conoscenza, le credenze, la possibilità, la necessità, eccetera.

Questo testo è stato pensato per lettori interessati principalmente alle applicazioni della logica all'intelligenza artificiale. L'enfasi costante è quindi al *linguaggio* della logica come formalismo di rappresentazione della conoscenza e agli *apparati deduttivi* come strumenti per l'automazione delle inferenze.

Trattiamo la logica proposizionale e la logica del primo ordine classiche, perché sono quelle che trovano più largo impiego in AI. Tra gli apparati deduttivi presentiamo il sistema hilbertiano e la deduzione naturale; il primo per la sua importanza storica, il secondo perché molto intuitivo. Accenniamo alla risoluzione – perchè storicamente ha avuto una notevole importanza, perchè è stata impiegata in molti dimostratori di teoremi e perchè soggiace al linguaggio di programmazione PROLOG – ma soprattutto trattiamo il metodo dei tableau, che all'eleganza formale aggiunge una maggiore leggibilità e intuitività.

Successivamente mostriamo alcune importanti applicazioni della logica in intelligenza artificiale.

Altre logiche per l'intelligenza artificiale, come pure gli aspetti legati a tecniche di implementazione di dimostratori automatici e di basi di conoscenza, non sono trattati nella presente versione di questo testo.

[3]Sistemi per la verifica automatica di hardware sono da tempo una realtà.

Organizzazione del testo

L'esposizione è divisa in tre parti.

Nella **prima parte** vengono introdotte le *nozioni di base*: dopo avere ricordato alcuni concetti fondamentali della teoria degli insiemi, della ricorsione, della calcolabilità, della decidibilità e della complessità computazionale, si passa a introdurre la nozione di *sistema formale* illustrandone la sintassi, la semantica e soffermandosi sulla nozione di deduzione.

Nella **seconda parte** vengono introdotte la *logica proposizionale* e la *logica del primo ordine*, ne vengono illustrati sintassi, semantica e decidibilità. Viene presentato l'apparato deduttivo assiomatico hilbertiano, introducendo le nozioni di completezza e correttezza dell'apparato deduttivo rispetto alla semantica. Viene poi introdotta la deduzione naturale. Particolare enfasi viene posta all'aspetto della deduzione automatica: vengono presentati il metodo dei tableau e la risoluzione, supportati — nel caso del calcolo dei predicati — dalle nozioni di unificazione e skolemizzazione.

Nella **terza parte** infine vengono presentate alcune applicazioni significative della logica in AI. Viene trattato il *ragionamento di senso comune* e il ragionamento in situazioni di conoscenza incompleta. Viene poi illustrato il ragionamento di tipo *abduttivo*. Viene illustrato il problema della diagnosi, sia con approccio abduttivo che basato su modelli. Viene poi illustrato il ragionamento su azioni e cambiamenti; in particolare viene introdotto il *calcolo delle situazioni* e si affronta il problema della *pianificazione* delle azioni di un robot;

Ogni capitolo si chiude con un riassunto delle nozioni più importanti introdotte; ogni paragrafo è corredato da molti esempi e si conclude con una serie di esercizi. Oltre a esercizi "classici", volti a favorire una migliore comprensione delle nozioni fondamentali, abbiamo inserito esercizi ispirati a problemi di intelligenza artificiale. Molti di questi sono ripresi esattamente – o adattati – da compiti d'esame del corso di Intelligenza Artificiale tenuto presso la Facoltà di Ingegneria dell'Università di Roma "La Sapienza".

Abbiamo deciso di *non* segnalare certi paragrafi come "opzionali" o certi esercizi come "difficili" in quanto i gusti e il background dei vari lettori possono variare.
Il testo si chiude con una ricca bibliografia, sia per approfondimenti sulle nozioni di base, sia per il lettore che voglia indirizzarsi alla ricerca nel settore e voglia costruirsi un quadro dello stato dell'arte.

Il testo può essere trattato in modo parziale: in un corso introduttivo di intelligenza artificiale, per esempio, possono essere omesse molte dimostrazioni e approfondimenti teorici e, tra le applicazioni di AI, se ne possono scegliere alcune a seconda dell'enfasi che si vuole dare al corso. Il materiale può essere trattato in modo più completo in un corso di "Logica per l'intelligenza artificiale".

Questa è la quarta versione del testo che, pur se molto aggiornata e ampliata rispetto alle versioni precedenti, non è ancora completa. Mancano dei capitoli e

altri abbisognano ancora di qualche emendamento; manca un indice analitico e la bibliografia va arricchita di altre voci.

Chi desideri fare osservazioni o fornire suggerimenti su questo testo ci farà cosa gradita se invierà un messaggio di posta elettronica a: {aiello, pirri}@dis.uniroma1.it.

Ringraziamenti

Innanzitutto ringraziamo gli studenti del corso di Intelligenza Artificiale del Corso di laurea in Ingegneria Informatica dell'Università di Roma "La Sapienza", che sono stati lettori attenti di una serie di versioni preliminari di questo testo. Un ringraziamento particolare a Gianfranco Piazzolla e Stefano Petrelli.

Ringraziamo Marco Benedetti e Marta Cialdea per avere riletto una precedente versione e per le future riletture.

Luigia Carlucci Aiello
Fiora Pirri

Roma, Marzo 2001

Contenuto

Parte I

Nozioni di Base

Introduciamo nel capitolo 1 le nozioni di insieme, funzione e relazione e di cardinalità degli insiemi. Accenniamo l'induzione e la ricorsione e la costruzione induttiva di insiemi. Presentiamo poi la nozione di calcolabilità e decidibilità e forniamo alcune nozioni introduttive di complessità computazionale.

Il capitolo 2 contiene la definizione di *logica* o *sistema formale*; ne vengono presentati: linguaggio, interpretazione e apparato deduttivo.

Capitolo 1
Preliminari

Questo capitolo ha la funzione di far familiarizzare il lettore con una terminologia e una notazione che saranno utilizzate ampiamente in seguito.

Iniziamo ricordando alcune nozioni sulla teoria degli insiemi e su relazioni, funzioni, operazioni, e loro proprietà. Per la maggior parte queste nozioni sono certamente note; si invita tuttavia a ripercorrerle: esse sono introduttive ai paragrafi successivi che trattano di cardinalità di insiemi, funzioni calcolabili e problemi di decisione. Introduciamo poi i principi di induzione matematica, l'induzione e la ricorsione su insiemi. Chiudiamo il capitolo presentando le nozioni di base della complessità computazionale.

1.1 Insiemi

Un insieme è una collezione di oggetti, detti elementi dell'insieme. Scriviamo $x \in S$ per dire che l'oggetto x è un elemento dell'insieme S e $x \notin S$ per dire che l'oggetto x non è elemento di S. $x = y$ sta a denotare che l'oggetto x è uguale all'oggetto y. Se S e T sono due insiemi e $S = T$, allora per ogni oggetto x, $x \in S$ sse $x \in T$; dove *sse* significa *se e solamente se*. Viceversa, *principio di estensionalità*:

> Se S e T sono due insiemi e per ogni oggetto x, $x \in S$ sse $x \in T$, allora $S = T$.

Osserviamo quindi che $\{x, y, x\} = \{x, y\}$, e che $\{x, y\} = \{y, x\}$, cioè in un insieme tutti gli elementi sono distinti e l'ordine in cui essi compaiono è irrilevante. L'insieme vuoto, ovvero l'insieme senza elementi, viene denotato con $\emptyset$. Per ogni oggetto x, esiste un insieme $\{x\}$ il cui unico elemento è proprio x; $\{x\}$ viene di solito detto *singoletto*. Più in generale, per ogni insieme finito di oggetti $x_1, \ldots, x_n$ esiste un insieme $\{x_1, \ldots, x_n\}$ i cui elementi sono esattamente tali oggetti.

Possiamo estendere tale notazione a insiemi infiniti. $\{0, 1, 2, \ldots\}$ è l'insieme dei naturali $\mathbb{N}$ e $\{\ldots, -1, 0, 1, \ldots\}$ è l'insieme degli interi relativi $\mathbb{Z}$.

Scriviamo $\{x|P(x)\}$ per indicare l'insieme di tutti gli oggetti x tali che $P(x)$ valga, ovvero tutti gli oggetti per i quali valgano certe asserzioni – o proprietà – che indichiamo con P. L'insieme di tali x è detto l'*estensione* di P.

Esempio 1

1. $\{x \mid x \in \mathbb{Z} \text{ e } x > 0\}$ *è l'insieme degli interi positivi.*
2. $\{x \mid x \in \{rosso, giallo, arancio\} \text{ e } x \text{ è un colore fondamentale}\}$ *è l'insieme* $\{rosso, giallo\}$.

Se S e T sono due insiemi e tutti gli elementi di S sono elementi di T diciamo che S è un *sottoinsieme* di T e scriviamo $S \subseteq T$; diciamo anche che T è un *soprainsieme* di S. Se $S \subseteq T$ e S è diverso da T diciamo che S è un *sottoinsieme proprio* di T e che T è un *soprainsieme proprio* di S e lo indichiamo con $S \subset T$. Ogni insieme è sottoinsieme di sé stesso e $\emptyset$ è sottoinsieme di ogni insieme.

L'insieme *potenza* di un insieme S, scritto $\wp S$ – detto anche *insieme delle parti* di S – è l'insieme formato da tutti i sottoinsiemi di S:

$$\wp S = \{X \mid X \subseteq S\}.$$

Esempio 2

1. $\wp\emptyset = \{\emptyset\}$
2. $\wp\{\emptyset\} = \{\emptyset, \{\emptyset\}\}$
3. $\wp\{x, y\} = \{\emptyset, \{x\}, \{y\}, \{x, y\}\}$.

L'*unione* di due insiemi S e T, scritto $S \cup T$, è l'insieme di tutti gli oggetti che sono elementi di S oppure di T. L'*intersezione* di due insiemi S e T, scritto $S \cap T$, è l'insieme di tutti gli oggetti che sono elementi sia di S che di T. S e T sono *disgiunti* se la loro intersezione è vuota.
Sia S un insieme i cui elementi sono insiemi, l'unione di tali insiemi è l'insieme:

$$\bigcup S = \{x : x \text{ appartiene a qualche elemento di } S\}$$

similmente, l'intersezione di tali insiemi è l'insieme

$$\bigcap S = \{x \mid x \text{ appartiene a tutti gli elementi di } S\}.$$

In particolare $\bigcup \wp S = S$. Nel caso in cui abbiamo un insieme S_n con $n \in \mathbb{N}$, scriviamo l'unione di tali insiemi $\bigcup_{n \in \mathbb{N}} S_n$ o, più semplicemente, $\bigcup_n S_n$. Useremo un'analoga notazione per l'intersezione.

Dati due insiemi X e Y con $X \subseteq Y$ chiamiamo $Y - X$ (scritto anche $Y \backslash X$) insieme *complemento* di X in Y. Se l'insieme Y può essere sottinteso, scriviamo anche $\overline{X}$. Ovviamente $(Y - X) \cap X = \overline{X} \cap X = \emptyset$ e $(Y - X) \cup X = \overline{X} \cup X = Y$.

Esempio 3

1. $\{x, y\} \subseteq \{x, y\}$
2. $\{x\} \subseteq \{x, y\}$
3. $\{x\} \subset \{x, y\}$
4. $\{x, y\} \cup \{y, z\} = \{x, y, z\}$
5. $\{x, y\} \cap \{y, z\} = \{y\}$
6. *Il complemento di* $\{x, y\}$ *in* $\{x, y, z\}$ *è* $\{z\}$.

Una *coppia ordinata* di oggetti x e y è definita in modo tale che

$$\langle x, y\rangle = \langle z, t\rangle \text{ sse } x = z \text{ e } y = t.$$

Notare che $\langle x, y\rangle \neq \langle y, x\rangle$ e che $\langle x, x\rangle$ denota la coppia in cui il primo e il secondo elemento sono uguali tra di loro.
Una *n-upla* ordinata di oggetti $x_1, \ldots, x_n$ è definita come:

$$\langle x_1, \ldots, x_n\rangle = \langle\langle x_1, \ldots, x_{n-1}\rangle, x_n\rangle$$

dove $\langle x_1, \ldots, x_{n-1}\rangle$ è una (n-1)-upla ordinata.

Diciamo che σ è una *sequenza finita* di elementi di T sse $\sigma = \langle s_1, \ldots, s_n\rangle$ per qualche intero positivo n e ciascun $s_i \in T$. Un *segmento* di una sequenza finita $\sigma = \langle s_1, \ldots, s_n\rangle$ è una sequenza finita $\sigma' = \langle s_k, s_{k+1} \ldots, s_{m-1}, s_m\rangle$, dove $1 \leq k \leq m \leq n$. Dati due insiemi S e T, possiamo formare l'insieme $S \times T$ di tutte le coppie $\langle x, y\rangle$ per le quali $x \in S$ e $y \in T$. L'insieme $S \times T$ è chiamato *il prodotto cartesiano* di S e T. S^n è l'insieme di tutte le n-uple di elementi di S, per esempio, per $n = 4$: $S^4 = (((S \times S) \times S) \times S)$.

1.1.1 Esercizi

Esercizio 1 *Dimostrare che se* S *è composto da* n *elementi (eventualmente* $n = 0$*), il numero di elementi in* $\wp S$ *è* 2^n.

Esercizio 2 *Verificare le seguenti proprietà:*

$$S \cap S = S \quad S \cap T = T \cap S$$

$$S \cap (T \cap R) = (S \cap T) \cap R$$

$$S \cap \emptyset = \emptyset \quad S \cup \emptyset = S$$

$$S \cap T \subseteq S \quad S \cap T \subseteq T$$

$$S \cap (T \cup R) = (S \cap T) \cup (S \cap R)$$

$$S \cup (T \cap R) = (S \cup T) \cap (S \cup R).$$

1.2 Relazioni, funzioni e operazioni

Una *relazione binaria* R tra due insiemi S e T è un insieme di coppie ordinate $\langle x, y\rangle$ con $x \in S$ e $y \in T$, cioè R è un sottoinsieme del prodotto cartesiano $S \times T$ (in simboli: $R \subseteq S \times T$). Il *dominio* di R è l'insieme di tutti gli oggetti x tali che $\langle x, y\rangle \in R$ per qualche y. Il *codominio* di R è l'insieme di tutti gli oggetti y tali che $y = f(x)$ per qualche x. L'unione del dominio e del codominio di una relazione R si chiama il *campo* di R o *estensione*.

Una relazione n-aria su un insieme S è un sottoinsieme di S^n, $n \geq 1$. In particolare, se $n = 1$ la relazione R su S si dice *unaria* ed è un sottoinsieme di S; se $n = 2$ la relazione R su S si dice *binaria* ed è un sottoinsieme di S^2, eccetera.

Talvolta le relazioni binarie vengono scritte con notazione infissa, cioè si scrive xRy invece di $\langle x, y\rangle \in R$.

Esempio 4

1. $\{\langle x, x\rangle \mid x \in S\}$ *è una relazione binaria su* S.
2. $\{\langle x, y\rangle \mid \langle x, y\rangle \in \mathbb{N}^2$ *e* x *è minore o uguale a* $y\}$ *è una relazione binaria su* $\mathbb{N}$ *e precisamente* la relazione d'ordine naturale *su* $\mathbb{N}$. *Essa viene scritta solitamente con notazione infissa, cioè come* $x \leq y$.
3. $\{\langle x, y, z\rangle \mid \langle x, y, z\rangle \in \mathbb{R}^3$ *e* $x^2 + y^2 = z^2\}$ *è una relazione ternaria sull'insieme dei reali* $\mathbb{R}$. *Tale relazione è chiamata il* luogo geometrico *dei punti in* $\mathbb{R}^3$ *che soddisfano l'equazione* $x^2 + y^2 = z^2$.

Una *funzione* f da un insieme S a un insieme T è una relazione $f \subseteq S \times T$ tale che per ogni x che appartiene al dominio di f esiste un unico y per cui $\langle x, y\rangle \in f$; tale unico y è detto il *valore* $f(x)$ che f assume in corrispondenza dell'argomento x. Se $x \in S$ è nel dominio di f allora si dice che $f(x)$ è definito, o anche che f è definita in x. Quindi una funzione f è definita per tutti gli elementi del suo dominio. Se il dominio di f coincide con S si dice che f è *totale*, altrimenti f è detta *parziale*.

Dati due insiemi S e T, se f è una *funzione* da S in T scriviamo

$$f : S \mapsto T$$

per indicare che il dominio di f è contenuto in S è e che il codominio di f è contenuto in T. In particolare $f(S) = \{y \mid y = f(x), x \in S\}$ si chiama (l'insieme) *immagine* di f, mentre l'insieme $f^{-1}(y) = \{x \mid y = f(x)\}$ si chiama (l'insieme) *immagine inversa* di f in y.

La nozione di immagine inversa si estende da un elemento a un sottoinsieme del codominio di una funzione f. Sia $f : S \mapsto T$, se V è un sottoinsieme di T, l'immagine inversa di V secondo f è $f^{-1}(V) = \{x \mid x \in S,$ e $f(x) \in V\}$, cioè $f^{-1}(V) = \bigcup_{y \in V} f^{-1}(y)$.

Se f è una funzione da S a T diremo in genere, che f *applica* S in T, oppure, usando un comodo inglesismo, che f *mappa* S in T.

Una funzione $f : S \mapsto T$ è *iniettiva* se per ogni $x, y \in S$ con $x \neq y$, risulta $f(x) \neq f(y)$, cioè se l'immagine inversa di ogni elemento del codominio di f è un insieme costituito da un solo elemento di S. Una funzione è *suriettiva* se per ogni $y \in T$ esiste un x in S tale che $f(x) = y$, in tal caso $f(S) = T$. Una funzione totale suriettiva e iniettiva è detta *biiettiva* o *biunivoca* o anche *uno-a-uno*.

Una funzione $f : S \mapsto T$ è detta *unaria* o *a un argomento*. La nozione di funzione si estende facilmente a più argomenti. Consideriamo $S \times S$, e $f : S \times S \mapsto T$; in questo caso f è detta *a due argomenti* o *binaria*. Se la coppia $\langle x_1, x_2 \rangle$ è nel dominio di f scriviamo $f(\langle x_1, x_2 \rangle) = f(x_1, x_2)$. Tale notazione si può estendere a n-uple: $f(\langle x_1, \ldots, x_n \rangle) = f(x_1, \ldots, x_n)$ per indicare $f : S_1 \times \ldots \times S_n \mapsto T$, dove $f(x_1, \ldots, x_n) = y$ con $x_i \in S_i$, $1 \leq i \leq n$, e $y \in T$.

Un'*operazione n-aria* su un insieme S è una funzione da S^n in S.

Esempio 5 *La divisione intera è una operazione binaria su $\mathbb{Z}$; essa è parziale perché non è definita per quelle coppie il cui secondo elemento è 0.*

Se f è una operazione binaria su S, essa si può anche scrivere con notazione infissa, cioè scriviamo $x_1 f x_2$ invece di $f(\langle x_1, x_2 \rangle)$ oppure di $f(x_1, x_2)$.
Se f è una operazione n-aria su S, la restrizione di f ad un sottoinsieme V di S è la funzione f_V avente dominio V^n e che concorda con f in ciascun punto di V^n:

$$f_V = f \cap (V^n \times S)$$

avremo $f_V(v_1, \ldots, v_n) = x$ sse $v_i \in V$ e $f(v_1, \ldots, v_n) = x$.

Una particolare operazione unaria su S è la funzione *identità* i definita come:

$$i(x) = x \qquad \text{per } x \in S.$$

Data una relazione binaria $R \subseteq S \times T$, la *relazione inversa* di R è definita come $R^{-1} = \{\langle y, x \rangle | \langle x, y \rangle \in R\}$. Pertanto, dalla definizione segue che $(R^{-1})^{-1} = R$.

Analogamente, data una funzione $f : S \mapsto T$, f ammette una *funzione inversa* $f^{-1} : T \mapsto S$ se f è iniettiva. Infatti, se la funzione è iniettiva le immagini inverse degli elementi di T sono insiemi costituiti da un solo elemento di S, per cui si può definire $f^{-1}(y) = x$.

Esempio 6

1. *La funzione doppio $f : \mathbb{N} \mapsto \mathbb{N}$, $f(x) = 2 \times x$ è iniettiva, ma non suriettiva.*
2. *La funzione $f : \mathbb{N} \mapsto \mathbb{N}$, $f(x) = x \div 10$ (dove $x \div y$ indica il quoziente della divisione tra x e y) non è iniettiva e neanche suriettiva.*
3. *La funzione $f : \mathbb{N} \mapsto \mathbb{N} - \{0\}$, $f(x) = x + 1$ è iniettiva e suriettiva, cioè è biunivoca, la funzione inversa f^{-1} essendo $f^{-1}(y) = y - 1$.*
4. *La funzione somma $f : \mathbb{N} \times \mathbb{N} \mapsto \mathbb{N}$, $f(x, y) = x + y$ non è iniettiva, quindi non è invertibile; infatti, dato un elemento $n \in \mathbb{N}$, esistono più coppie di numeri naturali $\langle x, y \rangle$ tali che $x + y = n$.*

Date due funzioni $f : S \mapsto T$ e $g : T \mapsto U$ la *composizione* di f e g è la funzione $f \circ g : S \mapsto U$ tale che $(f \circ g)(x) = g(f(x))$ per ogni $x \in S$. $(f \circ g)(x)$ è definita sse sono definite entrambe $g(f(x))$ e $f(x)$.

Esempio 7 *Se* $g(x) = 2 \times x$ *e* $f(x) = x + 3$ *allora* $(f \circ g)(x) = 2 \times (x + 3)$ *e* $(g \circ f)(x) = (2 \times x) + 3$.

1.2.1 Esercizi

Esercizio 3 *Sia* $f : S \mapsto T$, *e siano* $A, B \subseteq S$, *verificare se* $f(A \cap B) = f(A) \cap f(B)$ *e se* $f(A \cup B) = f(A) \cup f(B)$.

Esercizio 4 *Siano* $f : S \mapsto T$, $g : T \mapsto U$ *e* $h : U \mapsto Z$ *tre funzioni. Verificare che* $(f \circ (g \circ h))(x) = ((f \circ g) \circ h)(x)$. *Se* f *e* g *sono biiettive, mostrare che* $f \circ g$ *è biiettiva.*

1.3 Proprietà delle relazioni

Introduciamo ora alcune importanti proprietà delle relazioni.
Data una relazione binaria R su S diciamo che:
R è *riflessiva* sse $\langle x, x \rangle \in R$ per ogni $x \in S$;
R è *antiriflessiva* (o *irriflessiva*) sse $\langle x, x \rangle \notin R$ per ogni $x \in S$;
R è *simmetrica* sse $\langle x, y \rangle \in R$ qualora $\langle y, x \rangle \in R$;
R è *antisimmetrica* sse $\langle x, y \rangle \in R$ implica che $\langle y, x \rangle \notin R$;
R è *asimmetrica* sse non è simmetrica né antisimmetrica;
R è *transitiva* sse $\langle x, y \rangle \in R$ e $\langle y, z \rangle \in R$ comporta che $\langle x, z \rangle \in R$.

Esempio 8

1. *La relazione "essere sposati con" sull'insieme U degli esseri umani non è riflessiva, è simmetrica, non è transitiva.*
2. *La relazione "essere figlio di" sull'insieme U degli esseri umani non è riflessiva, non è simmetrica (è antisimmetrica), non è transitiva.*
3. *La relazione "essere avo di" sull'insieme U degli esseri umani non è riflessiva, non è simmetrica (è antisimmetrica), è transitiva.*
4. *La relazione "essere minore o uguale di" sull'insieme* $\mathbb{N}$ *è riflessiva, non è simmetrica (è asimmetrica), è transitiva.*

5. *La relazione "essere (strettamente) minore di" sull'insieme* $\mathbb{N}$ *non è riflessiva, non è simmetrica (è antisimmetrica), è transitiva.*

Definizione 1.1 *Una struttura relazionale* $\mathcal{SR}$ *è una n-upla in cui la prima componente è un insieme non vuoto* W *chiamato* universo *o* dominio *di* $\mathcal{SR}$ *e le cui rimanenti componenti sono relazioni (di arità varia) su* W*.*

Un *ordine parziale* o *semiordinamento* è una struttura relazionale data da una coppia $\langle S, R\rangle$ in cui S è un insieme ed R è una relazione binaria irriflessiva e transitiva su S.
Una relazione R di semiordinamento è un *ordinamento* sull'insieme S sse R è transitiva e per ogni $x, y \in S$ una e una sola delle tre condizioni seguenti è soddisfatta (questa proprietà è di solito detta *tricotomia*):

1. $x = y$;
2. $\langle x, y\rangle \in R$;
3. $\langle y, x\rangle \in R$.

In genere, un ordinamento è anche chiamato *ordine totale.*

Definizione 1.2 *Sia* $\langle S, R\rangle$ *una struttura relazionale con* R *relazione binaria su* S*.* R^+*, la* chiusura transitiva *di* R*, è la più piccola relazione transitiva su* S *che contiene* R*. Inoltre,* R^**, la* chiusura riflessiva e transitiva *di* R*, è la più piccola relazione riflessiva e transitiva su* S *che contiene* R*, cioè*

$$R^* = \bigcap\{R' \mid R' \textit{ relazione riflessiva e transitiva su } S \textit{ e } R \subseteq R'\}.$$

Si può dimostrare che, dati due elementi u e v di S, uR^*v sse esiste una sequenza finita di elementi $u = w_0, w_1 \ldots, w_n = v$ $(0 \leq n)$ di S tali che per ogni $i < n$ si abbia $w_i R w_{i+1}$. Quindi dire che uR^*v significa dire che v è raggiungibile da u in un numero finito di R-passi.

R è una relazione di *equivalenza* su un insieme S sse R è una relazione binaria su S che è riflessiva, simmetrica e transitiva.
Data una relazione di equivalenza R su un insieme S, la *classe di equivalenza* di un elemento $x \in S$ è definita come:

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

Dato un insieme S, una *partizione* di S è una famiglia di sottoinsiemi non vuoti di S, a due a due disgiunti e tali che la loro unione sia S.
Si può facilmente verificare che, data una relazione di equivalenza R in S, le classi di equivalenza generate da R *partizionano* S. In altri termini, le classi di equivalenza di S sono sottoinsiemi non vuoti di S tali che ogni elemento di S appartiene a una e una sola classe di equivalenza.

Si può altrettanto facilmente verificare che, data una qualunque partizione di un insieme S, essa induce una relazione di equivalenza su S; ponendo infatti due elementi di S in relazione R sse essi appartengono allo stesso elemento della partizione si ha che R è una relazione di equivalenza.

Data una relazione di equivalenza $\simeq$ in S, la partizione che essa determina dicesi l'*insieme quoziente* di S *rispetto a* $\simeq$, o anche *modulo* $\simeq$, e si indica con $S/\simeq$ o $S_{\simeq}$.

Esempio 9

1. *La relazione $<$ sull'insieme $\mathbb{N}$ è una relazione d'ordine totale.*
2. *Sia $\subset$ la relazione di contenimento proprio tra due insiemi. La relazione $\subset$ su $\wp\mathbb{N}$ è un ordine parziale.*
3. *La relazione $x \simeq_n y$ sui naturali definita come $x \simeq_n y$ sse $(x \bmod n) = (y \bmod n)$ (cioè se il resto della divisione intera tra x e n è lo stesso di quello della divisione intera tra y e n) è una relazione di equivalenza. Nel caso $n = 4$, le classi di equivalenza sono $[0], [1], [2]$ e $[3]$. L'insieme quoziente è $\{[0], [1], [2], [3]\}$, spesso indicato con $\mathbb{N}_4$.*

1.3.1 Esercizi

Esercizio 5 *Dare un esempio di relazione riflessiva sui naturali.*

Esercizio 6 *Dare un esempio di relazione simmetrica sui naturali.*

Esercizio 7 *Dare un esempio di relazione antisimmetrica sui naturali.*

Esercizio 8 *Dare un esempio di relazione asimmetrica sui naturali.*

Esercizio 9 *Dimostrare che se una relazione è simmetrica e transitiva allora è anche riflessiva.*

Esercizio 10 *Dire se la relazione "confinante con" tra due stati di un continente è una relazione di equivalenza.*

1.4 Cardinalità di insiemi

Due insiemi S e T si dicono *equipotenti* (e si scrive $S \sim T$) sse essi sono in corrispondenza biunivoca, cioè sse esiste una funzione biunivoca tra i due.

Un insieme S è *finito* se esiste una funzione biiettiva tra l'insieme S e l'insieme $\{0, 1, 2, \ldots, n\}$, per qualche n. Sia $N = \{1, 2, \ldots n\}$, se un insieme S è equipotente a N si dice che la *cardinalità* o *potenza* di S è n e si scrive: $|S| = |N| = n$. I numeri naturali n sono detti *numeri cardinali finiti*, gli altri cardinali sono detti *transfiniti*.

Esempio 10

1. *L'insieme dei giorni della settimana è finito e ha cardinalità* 7.
2. *L'insieme delle vocali dell'alfabeto dell'italiano è finito e ha cardinalità* 5.

La cardinalità dell'insieme $\mathbb{N}$ è denotata con $\aleph_0$ (alef con zero). $\aleph_0$ è il più piccolo cardinale transfinito. Ovviamente, per ogni cardinale finito n si ha che $n < \aleph_0$. Un insieme di potenza $\aleph_0$ è detto *numerabile*. Un insieme S è *numerabile* sse esiste una funzione biiettiva tra l'insieme S e l'insieme dei naturali $\mathbb{N}$.

Esempio 11

1. *L'insieme dei numeri pari è numerabile.*
2. *L'insieme dei numeri interi è numerabile.*
3. *Il prodotto cartesiano* $\mathbb{N} \times \mathbb{N}$ *è numerabile.*

I primi due esempi sono abbastanza intuitivi. Per provare che esiste una biiezione tra l'insieme $\{\langle n, m\rangle | n, m \in \mathbb{N}\}$, costruire i primi $k \times k$ elementi di una matrice infinita. Gli elementi della prima riga sono tutte le coppie $\langle 0, n\rangle$, $n \geq 0$, gli elementi della seconda riga saranno tutte le coppie $\langle 1, n\rangle$, $n \geq 0$ e così via. Costruire una biiezione con $\mathbb{N}$ come segue $f(\langle 0, 0\rangle) = 0$, $f(\langle 0, 1\rangle) = 1$, $f(\langle 1, 0\rangle) = 2$, $f(\langle 0, 2\rangle) = 3$, $f(\langle i, j\rangle) = i + \dfrac{(i + j) \times (i + j + 1)}{2}$.

Quanto abbiamo visto precedentemente, si può generalizzare con il seguente teorema:

Teorema 1.3 *Sia S un insieme finito o numerabile. L'insieme $\mathcal{S}$ di tutte le sequenze finite di elementi di S è anch'esso numerabile.*

Dimostrazione Lasciata al lettore (si veda l'esercizio 11). Si ricorda che l'insieme di tali sequenze può essere definito come

$$\mathcal{S} = \bigcup_{n \in \mathbb{N}} S^{n+1}$$

Poiché S è finito o numerabile esiste una funzione biiettiva che porta S in $\mathbb{N}$ o in un suo sottoinsieme finito. Basta quindi trovare una funzione che porta sequenze di $\mathcal{S}$ in $\mathbb{N}$. L'insieme $\mathcal{S}$ così costruito viene spesso indicato con S^* dove $*$ è detto "operatore stella di Kleene".

D'altro canto esiste una gerarchia infinita di insiemi non numerabili. Si può dimostrare che se $|S| = \sigma$, $|\wp S| = 2^\sigma$ e che $\sigma < 2^\sigma$ per ogni σ finito o transfinito. Se $\sigma = \aleph_0$, $2^{\aleph_0}$ viene chiamato la *cardinalità del continuo* e denotato con $\mathbf{c}$ oppure con $\aleph_1$. Una ipotesi comunemente accettata in matematica è che non esistano cardinalità intermedie tra $\aleph_0$ e $2^{\aleph_0} = \mathbf{c}$, questa è detta *ipotesi del continuo*.

La tecnica di dimostrazione che consiste nel tentare di costruire una biiezione tra un insieme X supposto non numerabile e $\mathbb{N}$ e nel verificare che qualche elemento di X sfugge alla biiezione è stata inventata da Cantor per dimostrare la non numerabilità dei reali (e cioè che $2^{\aleph_0}$ è strettamente superiore ad $\aleph_0$) e si chiama tecnica di *diagonalizzazione*.

Dimostriamo, usando il metodo di Cantor, che $\wp\mathbb{N}$ non è numerabile.

L'idea della diagonalizzazione è la seguente. Supponiamo che si possa stabilire una corrispondenza biunivoca tra $\mathbb{N}$ e $\wp\mathbb{N}$. Questa ci permette di enumerare tutti gli elementi di $\wp\mathbb{N}$: X_0, X_1, X_2,..., X_n, ... Costruiamo un insieme Y scegliendo per ciascun X_i un elemento che non appartiene a X_i, sia esso n_i. Così prendiamo n_0 che non appartiene a X_0, n_1 che non appartiene a X_1, eccetera. Y chiaramente appartiene a $\wp\mathbb{N}$, ma per costruzione non è in corrispondenza con alcun n perché differisce da ciascun X_n almeno per l'elemento x_n. Questo mostra che c'è un elemento di $\wp\mathbb{N}$ che è sfuggito alla numerazione, quindi $\wp\mathbb{N}$ non è in corrispondenza biunivoca con $\mathbb{N}$.

1.4.1 Esercizi

Esercizio 11 *Dimostrare it teorema 1.3*

Esercizio 12 *Dimostrare che se* $|S| = n$ *allora* $|\wp S| = 2^n$, *per ogni naturale* n.

Esercizio 13 *Dimostrare che la cardinalità dell'insieme dei numeri primi è* $\aleph_0$.

Esercizio 14 *Dimostrare che* $\aleph_0 + n = \aleph_0$; $\aleph_0 + \aleph_0 = \aleph_0$; $\aleph_0 \times n = \aleph_0$; $\aleph_0 \times \aleph_0 = \aleph_0$.

1.5 Principi di induzione matematica

Se C è il campo di una relazione R su un insieme X, sia S un sottoinsieme di C; un elemento x di S si dice *R-minimo* in S se $\langle x, z\rangle \in R$ per ogni $z \in S$ diverso da x. Un *buon ordinamento* è un ordinamento tale che ogni sottoinsieme non vuoto del campo di R ha un elemento R-minimo.

[ASSIOMA DEL BUON ORDINAMENTO]
Ogni sottoinsieme non vuoto dei numeri naturali possiede un elemento minimo.

In altre parole, se $\emptyset \neq S \subseteq \mathbb{N}$, allora esiste un m, $m \in S$ tale che $m < n$ per ogni $n \in S$ (ovvero m è R-minimo, dove R è la relazione $<$). Abbiamo utilizzato il termine *assioma* perché assumiamo, in questo contesto, tale proprietà come primitiva. Da essa si possono dimostrare altre proprietà. Una proprietà che si deriva è l'induzione:

Teorema 1.4 [PRINCIPIO DI INDUZIONE MATEMATICA] *Sia $A(n)$ una asserzione sull'insieme dei naturali $\mathbb{N}$ e supponiamo che:*

1. *$A(0)$ è vera;*
2. *per ogni $k \in \mathbb{N}$, se è vera $A(k)$ allora è vera $A(k+1)$.*

Allora $A(n)$ è vera per ogni $n \in \mathbb{N}$.

Dimostrazione Sia S l'insieme dei naturali per cui l'asserzione è falsa, cioè $S = \{x | x \in \mathbb{N}, A(x) \text{ è falsa}\}$. Vogliamo dimostrare che S è vuoto. Supponiamo che sia $S \neq \emptyset$, mostreremo che ne segue una contraddizione. Per l'assioma del buon ordinamento S ha un elemento minimo s e, per definizione di S, $A(s)$ deve essere falsa. Per l'ipotesi 1, $A(0)$ è vera, dunque $s > 0$, siccome s è il minimo di S, $(s-1) \notin S$ e perciò $A(s-1)$ è vera. Ma, per 2, se $A(s-1)$ è vera anche $A((s-1)+1) = A(s)$ è vera. Contraddizione. □

Teorema 1.5 [PRINCIPIO DI INDUZIONE COMPLETA] *Sia $A(n)$ una asserzione sull'insieme dei naturali $\mathbb{N}$ e supponiamo che:*

1. *$A(0)$ è vera;*
2. *per ogni $m \in \mathbb{N}$, $m > 0$, se $A(k)$ è vera per ogni k, $0 \leq k < m$, ne segue che è vera $A(m)$.*

Allora $A(n)$ è vera per ogni $n \in \mathbb{N}$, $n \geq 0$.

Dimostrazione Sia S l'insieme dei naturali x per cui l'asserzione $A(x)$ è falsa. Similmente alla precedente dimostrazione, supponiamo $S \neq \emptyset$ e sia s il minimo. Osserviamo che per l'ipotesi 1, $A(0)$ è vera, dunque $s > 0$, e $A(k)$ è vera per ogni $k < s$, pertanto da 2 segue che $A(s)$ è vera. Contraddizione. □

Mostriamo ora un problema la cui dimostrazione necessita dell'induzione matematica nella seconda forma, ovvero l'induzione completa:

Esempio 12 *Siano n, m due interi con $m > 0$ e $n \geq 0$. Esistono due interi q, r con $0 \leq r < m$, tali che $n = m \times q + r$.*

Dimostrazione Per induzione completa. Dati m ed n, dobbiamo trovare q ed r per cui $n = mq + r$.
(*Passo Base*) Se $n = 0$ allora l'asserto $A(0)$ è verificato con $q = r = 0$.
(*Passo Induttivo*) Sia $n > 0$, dobbiamo verificare che $A(n)$ segue dall'ipotesi che $A(k)$ è vero per ogni k, $0 \leq k < n$. Se $m > n$ è sufficiente porre $q = 0$ e $r = n$. Se $m \leq n$ allora $0 \leq n - m < n$ e per ipotesi induttiva $A(n - m)$ è vero. Dunque esistono due numeri q' ed r' tali che $n - m = m \times q' + r'$, $0 \leq r' < m$. Allora $n = m + m \times q' + r' = m \times (1 + q') + r'$. Dunque abbiamo dimostrato $A(n)$ e, per il principio di induzione, $A(n)$ è vera per ogni n. □

1.5.1 Esercizi

Esercizio 15 *Dimostrare che la somma dei primi n dispari è uguale a n^2. Suggerimento: usare l'induzione.*

Esercizio 16 *Dimostrare che:*

$$(1 + x)^n = \binom{n}{0} + \binom{n}{1} x + \binom{n}{2} x^2 + \ldots + \binom{n}{n} x^n$$

Usare l'induzione.

1.6 Induzione e ricorsione su insiemi

La tecnica di dimostrazione per induzione sui naturali, che abbiamo ricordato nel precedente paragrafo, è ampiamente nota: per provare un asserto A, o una certa proprietà A, relativa ai naturali, si prova $A(0)$ e quindi si prova che se è vero $A(n)$ allora è vero $A(n + 1)$. In effetti, tale tecnica di ragionamento si basa sul modo in cui l'insieme dei naturali è definito: il principio di induzione matematica ricalca la "struttura" dei naturali. I naturali sono generati da un insieme iniziale, $\{0\}$, per mezzo di una operazione successore, s:

$$\mathbb{N} = \{0, s(0), s(s(0)), s(s(s(0))), \ldots\}.$$

Quando stabiliamo l'asserto A per induzione, sfruttiamo il fatto che un generico numero naturale n è ottenuto da 0 applicando n volte l'operatore s. E, quindi, applicare il principio di induzione per dimostrare che una proprietà vale sui naturali significa dire che se la proprietà vale per lo zero e si conserva attraverso il passaggio da un numero al suo successore, allora vale per tutti i naturali. E, infatti, tutti (e soli) i naturali si ottengono a partire dallo 0 mediante ripetute applicazioni dell'operatore successore.

In altri termini, è proprio la definizione dell'insieme dei naturali, data da Peano, che permette il ragionamento induttivo. L'essenza di tale definizione è che essa è allo stesso tempo una tecnica di costruzione dell'insieme $\mathbb{N}$ e una tecnica di prova.

Spesso, in informatica e intelligenza artificiale ci troviamo a definire degli insiemi infiniti fornendo le regole di costruzione dei relativi elementi, a partire da alcuni elementi considerati primitivi.

Se vogliamo concepire una costruzione per un insieme qualunque, in qualche modo simile a quella dei naturali, che ci permetta di effettuare dimostrazioni per induzione sull'insieme costruito possiamo fornire due *metodi di definizione*: uno *top-down* ed uno *bottom-up*. Vedremo (lemma 1.6) che i due metodi sono di fatto equivalenti.

Dato un insieme U, supponiamo di voler costruire un sottoinsieme di U a partire da alcuni elementi iniziali di U, applicando certe operazioni. L'insieme che vogliamo costruire deve contenere tutti gli elementi iniziali e tutti e soli gli elementi che si possono ottenere da essi applicando ripetutamente tali operazioni.

Sia $\Im$ l'insieme di elementi iniziali ($\Im \subseteq U$) e sia $\mathcal{F}$ l'insieme di operazioni su U. Supponiamo che le uniche due operazioni in $\mathcal{F}$ siano:

$$f : U \times U \mapsto U \qquad\qquad g : U \mapsto U$$

vogliamo definire l'insieme, che chiamiamo C che contiene tutto e solamente quello che possiamo costruire tramite f e g dall'insieme iniziale $\Im$.

Osserviamo che l'insieme delle funzioni non deve essere necessariamente finito, e neppure l'insieme iniziale. L'ipotesi che qui facciamo non è restrittiva, ci serve solo per semplicità. In effetti, abbiamo scelto espressamente una funzione unaria e una binaria perché gli insiemi che andremo a costruire successivamente, saranno caratterizzati dalla presenza di operazioni unarie e binarie.

Mostriamo, prima di tutto, come si definisce l'insieme C secondo un metodo top-down.

(*top-down*) Diciamo che un insieme $S \subseteq U$ è *chiuso* rispetto a f e g sse ogni qual volta $x, y \in S$ allora sia $g(x)$ che $f(x, y)$ sono in S. Un insieme S è detto *induttivo* su $\Im$ mediante le funzioni di $\mathcal{F}$ (cioè f e g) sse $\Im \subseteq S$ ed S è chiuso rispetto a f e g.

Sia $C^* = \bigcap S$ con S induttivo su $\Im$ mediante f e g; abbiamo che $x \in C^*$ sse x appartiene a tutti i sottoinsiemi induttivi di U. Ce ne deve essere almeno uno, per esempio U.

Chiaramente C^* è il più piccolo insieme che contiene $\Im$ ed è chiuso rispetto a f e g.

(*bottom-up*) Vogliamo costruire l'insieme C_* a partire dall'insieme iniziale $\Im \subset U$, applicando le operazioni f e g un numero finito (ma arbitrario) di volte. Definiamo una sequenza costruttiva di sottoinsiemi C_i di U, come segue:

$$C_0 = \Im$$

$$C_{i+1} = C_i \cup \{f(x, y) | x, y \in C_i\} \cup \{g(x) | x \in C_i\}.$$

È chiaro che $C_i \subseteq C_{i+1}$, per ogni $i \geq 0$. Definiamo, ora

$$C_* = \bigcup_{i \geq 0} C_i.$$

Lemma 1.6
$C^* = C_*$.

Dimostrazione Per provare $C^* = C_*$ proviamo separatamente $C^* \subseteq C_*$ e $C_* \subseteq C^*$.

Per provare che $C^* \subseteq C_*$ dobbiamo mostrare che C_* è induttivo, ovvero che contiene $\Im$ (ma questo è ovvio perché $C_0 \subseteq C_*$) e che è chiuso rispetto a f e g. Supponiamo $x \in C_*$, pertanto $x \in C_i$, per qualche $i \geq 0$. Per definizione di C_{i+1}, $g(x) \in C_{i+1} \subseteq C_*$ e, pertanto, $g(x) \in C_*$. Un ragionamento analogo si può fare per $f(x, y)$. Quindi C_* è induttivo e, siccome C^* è per costruzione il più piccolo insieme induttivo, $C^* \subseteq C_*$.

Per provare che $C_* \subseteq C^*$, occorre provare per induzione su i che $C_i \subseteq C^*$, per ogni i. Chiaramente $\Im = C_0 \subseteq C^*$ perché C^* è induttivo. Assumiamo, per ipotesi induttiva, che $C_i \subseteq C^*$; supponiamo $x, y \in C_i \subseteq C^*$, siccome C^* è induttivo allora $g(x), f(x, y) \in C^*$ e, dunque, per definizione di C_{i+1}, $C_{i+1} \subseteq C^*$.

Abbiamo, pertanto, mostrato che

$$\bigcup_{i \geq 0} C_i = C_* = C^* = \bigcap \{S \mid S \text{ induttivo su } \Im \text{ mediante } f \text{ e } g\}.$$

□

Sia $C = C^* = C_*$. Chiamiamo C l'*insieme generato* da $\Im$ per mezzo delle funzioni in $\mathcal{F}$. Chiaramente l'insieme C contiene tutti e soli gli elementi che si possono costruire a partire dall'insieme iniziale $\Im$ per mezzo di f e g:

Teorema 1.7 [PRINCIPIO DI INDUZIONE PER INSIEMI INDUTTIVI] *Sia C l'insieme generato da $\Im$ per mezzo delle funzioni in $\mathcal{F}$. Se S è un sottoinsieme di C che contiene $\Im$ ed è chiuso rispetto alle funzioni in $\mathcal{F}$, allora $S = C$.*

Dimostrazione Per ipotesi S è induttivo. Dunque, $C^* \subseteq S$ e per il lemma 1.6 $C \subseteq S$. Siccome per ipotesi $S \subseteq C$, ne segue che $C = S$. □

Spesso ci troveremo a definire delle funzioni *ricorsivamente* su degli insiemi induttivi. Date certe regole costruttive per gli elementi dell'insieme vogliamo definire delle funzioni basandoci sulla costruzione fatta. Ovvero vogliamo poter specificare le funzioni in base al valore che esse assumono sugli elementi iniziali e specificando il comportamento delle funzioni sugli elementi ottenuti per costruzione.

Esempio 13 *Quale illustrazione di definizione ricorsiva che ricalca la struttura induttiva dei numeri naturali consideriamo la seguente definizione della funzione fattoriale:*

1. $fatt(0) = 1$,

2. $fatt(s(n)) = s(n) \times fatt(n)$ *per ogni* n *positivo.*

Il prossimo esempio mostra come, usando lo schema seguito per la definizione del fattoriale su un insieme costruito a partire da un generico $\Im$ ed $\mathcal{F}$, non necessariamente si pervenga alla definizione di una funzione.

Esempio 14 *Sia* $U = \mathbb{R}$*, l'insieme dei numeri reali e siano:*

1. $\Im = \{0\}$,
2. $f(x, y) = x \times y$,
3. $g(x) = x + 1$.

L'insieme C generato da $\Im$, e chiuso rispetto alle operazioni f e g, è l'insieme dei naturali. Consideriamo una funzione h costruita *ricorsivamente* nel seguente modo:

1. $h(0) = 0$;
2. $h(f(x, y)) = f(h(x), h(y))$;
3. $h(g(x)) = h(x) + 2$.

Una simile funzione h non esiste, infatti $1 = g(0)$, quindi:

$$h(1) = h(g(0)) = h(0) + 2 = 2$$

tuttavia, abbiamo anche che: $1 = f(1, 1)$ e, quindi,

$$h(1) = h(f(1, 1)) = h(f(g(0), g(0))) = f(h(g(0)), h(g(0))) = f(2, 2) = 4$$

□

Introduciamo la nozione di insieme liberamente generato.

Definizione 1.8 *Un insieme* C *è liberamente generato da* $\Im$ *per mezzo delle operazioni in* $\mathcal{F}$ *se è generato e inoltre:*

1. *Le funzioni* f *e* g *in* $\mathcal{F}$*, ristrette a* C*,* f_C *e* g_C*, sono biiettive;*
2. *Il codominio di* f_C*, il codominio di* g_C *e l'insieme* $\Im$ *sono a due a due disgiunti.*

Osserviamo che, nel precedente esempio, C non è liberamente generato.

Teorema 1.9 [TEOREMA DI RICORSIONE] *Sia* C *un sottoinsieme di* U*, liberamente generato da* $\Im \subseteq U$ *per mezzo di* f *e* g*, dove* $f : U \times U \mapsto U$ *e* $g : U \mapsto U$*. Inoltre assumiamo che* V *sia un insieme qualunque, e* F*,* G *e* h *funzioni tali che:* $h : \Im \mapsto V$*,* $F : V \times V \mapsto V$ *e* $G : V \mapsto V$*. Esiste ed è unica la funzione* $\overline{h} : C \mapsto V$ *che estende* h *e tale che:*

1. $\overline{h}(x) = h(x)$*, per* $x \in \Im$*;*

2. $\overline{h}(f(x,y)) = F(\overline{h}(x), \overline{h}(y))$, *per* $x, y \in C$,
3. $\overline{h}(g(x)) = G(\overline{h}(x))$, *per* $x \in C$

In altri termini il teorema dice che la funzione h da $\Im$ nell'insieme V può essere estesa in modo unico a una funzione dall'insieme liberamente generato C, su cui abbiamo le operazioni f e g, all'insieme V su cui abbiamo delle corrispondenti operazioni F e G in modo che commutino i seguenti diagrammi.

$$\begin{array}{ccc} C \times C & \xrightarrow{f} & C \\ {\scriptstyle \langle \overline{h}, \overline{h} \rangle}\downarrow & & \downarrow{\scriptstyle \overline{h}} \\ V \times V & \xrightarrow{F} & V \end{array}$$

$$\begin{array}{ccc} C & \xrightarrow{g} & C \\ {\scriptstyle \overline{h}}\downarrow & & \downarrow{\scriptstyle \overline{h}} \\ V & \xrightarrow{G} & V \end{array}$$

Diamo solo una traccia di questa dimostrazione, il lettore può completarla come esercizio (si veda l'esercizio 17).

1. Siano $V' \subseteq V$ e $C' \subseteq C$, costruiamo delle funzioni $v : C' \mapsto V'$ tali che:

   i. per ogni elemento x che appartiene sia all'insieme degli elementi iniziali; $\Im$ che a C' abbiamo $v(x) = h(x)$

   ii. se $f(x,y)$ appartiene a C' allora anche x e y appartengono a C' e, inoltre, $v(f(x,y)) = F(v(x), v(y))$. Analogamente, se $g(x)$ appartiene a C', allora x appartiene a C' e $v(g(x)) = G(v(x))$.

   Sia Q l'insieme di tali funzioni v al variare di V' e C'. Si può dimostrare che Q non è vuoto. Definiamo $\overline{h}$ come

   $$\langle x, y \rangle \in \overline{h} \text{ sse } v(x) = y \text{ per qualche } v \in Q.$$

2. Per dimostrare che $\overline{h}$ soddisfa le condizioni richieste dal teorema, è sufficiente mostrare che:

   (a) $\overline{h}$ è una funzione;
   (b) $\overline{h}$ ha le caratteristiche delle funzioni v;
   (c) $\overline{h}$ è definita su tutto C;
   (d) $\overline{h}$ è unica.

□

1.6.1 Esercizi

Esercizio 17 *Completare la dimostrazione del teorema 1.9.*

Esercizio 18 *Sia A un insieme, $B \subseteq A$ un sottoinsieme di A ed $F = \bigcup_{n\in\mathbb{N}} F_n$ sia l'insieme delle funzioni tali che, per ciascun $f \in F_n$, $f : A^n \mapsto A$ è una funzione n-aria. Un insieme $X \subseteq A$ è F-chiuso rispetto a B se:*

1. *$B \subseteq X$;*
2. *per ogni $n \in \mathbb{N}$, ogni $f \in F$ e ogni n-upla $\langle a_1, \ldots, a_n\rangle \in X^n$ anche $f(a_1, \ldots, a_n) \in X$.*

Dimostrare che:

1. *A è F-chiuso per ogni $B \subseteq A$;*
2. *Siano $X, Y \subseteq A$ F-chiusi rispetto a qualche $B \subseteq A$. Allora $X \cap Y$ è F-chiuso rispetto a B.*
3. *Sia $\mathcal{X}$ un insieme di insiemi $X \subseteq A$ che sono F-chiusi rispetto a qualche $B \subseteq A$. Allora $\bigcap \mathcal{X}$ è F-chiuso rispetto a B.*

Esercizio 19 *Siano A, B ed F come nel precedente esercizio. Sia $X_{B,F}$ l'intersezione di tutti i sottoinsiemi di A che sono F-chiusi rispetto a B. Si dimostri:*

1. *$X_{B,F}$ è F-chiuso rispetto a B.*
2. *Se $X \subseteq A$ è F-chiuso rispetto a B allora $X_{B,F} \subseteq X$. In altri termini, $X_{B,F}$ è il più piccolo sottoinsieme di A che è F-chiuso rispetto a B.*

Esercizio 20 *Sia fatt la funzione definita nell'esempio 13. Dimostrare che per ogni $n \in \mathbb{N}$ con $n > 3$ si ha: $2^n - 1 \leq$ fatt(n).*

1.7 Funzioni calcolabili e modelli di calcolo

In questo paragrafo discutiamo succintamente le funzioni calcolabili e i modelli di calcolo[1]. Essi saranno importanti nei successivi capitoli in quanto ci permettono di introdurre la nozione di decidibilità.

Consideriamo la funzione *successore* s che abbiamo visto nei paragrafi precedenti e introduciamo i seguenti fatti:

1. $s^n(0) = \underbrace{s(s \ldots s(0) \ldots)}_{n} = n$ per ogni $n > 0$

[1] Per entrambi gli argomenti uno studio approfondito si trova nel libro di Rogers [44].

2. $s^n(0) \neq 0$, per ogni $n > 0$
3. $s(s^n(0)) = s(s^m(0))$ sse $s^n(0) = s^m(0)$, per ogni n ed m.

Notiamo che 1, 2 e 3 sono schemi. Consideriamo l'addizione così definita:

4. $s^n(0) + 0 = s^n(0)$
5. $s^n(0) + s(s^m(0)) = s(s^n(0) + s^m(0))$

Con le definizioni fornite sopra, eventualmente utilizzando le proprietà dell'uguaglianza[2] mostriamo che possiamo derivare:

$$3 + 2 = 5$$

(a)	$s(s(s(0))) = s^3(0) = 3$	per 1
(b)	$s(s(0)) = s^2(0) = 2$	per 1
(c)	$3 + 2 = s(s(s(0))) + s(s(0))$	da a, b
(d)	$s(s(s(0))) + s(s(0)) = s(\ s(s(s(0))) + s(0))$	per 5
(e)	$s(\ s(s(s(0))) + s(0)) = s(\ s(\ s(\ s(s(0))) + 0)\ )$	per 5 e 3
(f)	$s(\ s(\ s(s(s(0))) + 0)) = s(\ s(s(s(s(0)))\ )\ )$	per 4 e 3
(g)	$s(\ s(\ s^3(0))) = s^5(0) = 5$	da f, per 1.

Abbiamo calcolato 3+2 usando le definizioni fornite sopra e applicandole opportunamente seguendo quella che può essere definita una deduzione formale. Supponiamo ora di avere una macchina addizionatrice che, prendendo in input due qualunque numeri naturali, ne fornisce la somma. La macchina potrebbe utilizzare le definizioni fornite sopra, oppure potrebbe utilizzare altre definizioni non meglio specificate. Comunque sia ci aspettiamo che la macchina esegua una sequenza di istruzioni che non richiedano alcuna particolare abilità. L'esecuzione di una sequenza finita di istruzioni che non richiedono alcuna particolare abilità e che fornisce in output un risultato, si chiama *procedura effettiva*. Ricordiamo che una procedura effettiva A, o metodo effettivo, è definita come segue:

1. A è un insieme finito e non ambiguo di istruzioni, dove ogni istruzione è espressa in un insieme finito di simboli.
2. A, se eseguita senza errori, fornisce sempre il risultato desiderato in un numero finito di passi.
3. A può essere eseguita da una persona solo con l'ausilio di carta e penna.
4. A non richiede alcuna abilità specifica per essere eseguita.

Relativamente al punto 1, dato un insieme W di simboli, a partire da W si può definire l'insieme di istruzioni come un insieme induttivo (si veda l'esercizio 22). L'insieme delle istruzioni viene chiamato il *linguaggio* in cui è espresso A. Una procedura effettiva è dunque un algoritmo espresso in un linguaggio dato.

[2] Si veda il paragrafo 1.3.

Sia $\mathcal{U}$ un linguaggio in cui può essere espressa una procedura effettiva (o algoritmo). Data una funzione $f : \mathbb{N} \mapsto \mathbb{N}$ diciamo che f *è calcolabile in* $\mathcal{U}$ sse esiste un algoritmo A espresso in $\mathcal{U}$ che la calcola; cioè un algoritmo che termina su un input n se f è definita su n e fornisce in output $f(n)$. In particolare, una funzione calcolata da un algoritmo si dice *calcolabile* o *effettivamente calcolabile*.

Si noti che mentre a un algoritmo si associa una unica funzione calcolata, a una funzione da $\mathbb{N}$ a $\mathbb{N}$ può essere in genere associato più di un algoritmo che la calcola. Sia $A_{\mathcal{U}}$ l'insieme di tutti i possibili algoritmi in $\mathcal{U}$ e sia

$$Fun_{\mathcal{U}} = \{f_A | A \in A_{\mathcal{U}}\}$$

$Fun_{\mathcal{U}}$ è detto l'insieme di tutte le funzioni calcolabili in $\mathcal{U}$.

La cardinalità di $Fun_{\mathcal{U}}$ è la misura del potere espressivo di $\mathcal{U}$, cioè la quantità di funzioni che si possono calcolare con algoritmi di $\mathcal{U}$. Poniamoci la seguente domanda: può esistere un linguaggio $\mathcal{U}$ tale che l'insieme $Fun_{\mathcal{U}}$ ad esso associato coincide con l'insieme di tutte le funzioni (dai naturali ai naturali)? Se la risposta fosse sì potremmo concludere che tutte le funzioni sono calcolabili (in un opportuno linguaggio). Invece si può facilmente dimostrare che la risposta è negativa.

Infatti, dato un linguaggio $\mathcal{U}$, essendo gli algoritmi delle sequenze finite di istruzioni di $\mathcal{U}$, essi sono in quantità al più numerabile (si veda il teorema 1.3), quindi $|Fun_{\mathcal{U}}| \leq \aleph_0$. Viceversa si può dimostrare, con un ragionamento diagonale, che l'insieme $Fun_{\mathbb{N},\mathbb{N}}$ di tutte le funzioni dai naturali ai naturali

$$Fun_{\mathbb{N},\mathbb{N}} = \{f \mid f : \mathbb{N} \mapsto \mathbb{N}\}$$

ha cardinalità $2^{\aleph_0}$. Infatti, supponiamo $Fun_{\mathbb{N},\mathbb{N}}$ numerabile; sia $f_1, f_2, \ldots f_n, \ldots$ una numerazione. Sia $g(m) = f_m(m) + 1$. Quindi $g \in Fun_{\mathbb{N},\mathbb{N}}$, per cui esiste un numero naturale k tale che $g = f_k$, cioè g sarà la k-esima funzione della numerazione, per qualche k. Contraddizione in quanto $g(k) = f_k(k) = f_k(k) + 1$.

Il problema che si sono posti i logici nella prima metà del '900 è stato quello di caratterizzare le funzioni calcolabili.

Consideriamo un sottoinsieme particolare delle funzioni dai naturali ai naturali, le *funzioni ricorsive*. Ricordiamo che una funzione F si dice *ricorsiva* se $F : \mathbb{N}^n \mapsto \mathbb{N}$ ed è ottenuta per mezzo dei seguenti schemi:

(1) $I_i^n(a_1, \ldots, a_n) = a_i$
$a_1 + a_2$
$a_1 \times a_2$
$G_<(a_1, a_2) \begin{cases} = 1 & \text{se } a_1 < a_2 \\ = 0 & \text{altrimenti} \end{cases}$
sono funzioni ricorsive.

(2) Se G è una funzione ricorsiva a k argomenti e
$H_1 \cdots H_k$ sono funzioni ricorsive a n argomenti, la funzione:
$F(a_1, \ldots, a_n) = G(H_1(a_1, \ldots, a_n), \ldots, H_k(a_1, \ldots, a_n))$
è ricorsiva.

(3) Se G è ricorsiva e per ogni sequenza $a_1, \ldots, a_n$ esiste un b tale che:
$G(a_1, \ldots, a_n, b) = 0$
allora la funzione : $F(a_1, \ldots, a_n) = \mu x(G(a_1, \ldots, a_n, x) = 0)$
è ricorsiva, dove $\mu x(G(a_1, \ldots, a_n, x) = 0)$ denota il più
piccolo x tale che $G(a_1, \ldots, a_n, x) = 0$.

Gli schemi forniti sopra per le funzioni ricorsive costituiscono un *modello di calcolo* formale. Qual è, dunque, la relazione fra funzioni calcolabili tramite una procedura effettiva e funzioni ricorsive? Se F è una funzione ricorsiva allora è chiaro che F è calcolabile. Per esempio, se consideriamo il punto 3 sopra definito per le funzioni ricorsive, data una sequenza qualunque $a_1, \ldots, a_n$ di numeri, possiamo definire un algoritmo che verifica

$$G(a_1, \ldots, a_n, x) = 0$$

provando prima con $x = 0$, poi con $x = 1$ e così via, finché non viene trovato il primo valore per x tale che $G(a_1, \ldots, a_n, x) = 0$.

Come possiamo stabilire il viceversa, ovvero risalire da una procedura effettiva a una definizione ricorsiva?

Nel 1936 Church propose la seguente ipotesi di lavoro:

> [TESI DI CHURCH, 1936] Qualunque funzione sui naturali effettivamente calcolabile è ricorsiva.

Nella tesi solo la nozione di "funzione ricorsiva" ha una definizione rigorosa, mentre la nozione di "funzione effettivamente calcolabile" si riferisce a un concetto di calcolo non precisato matematicamente, ma effettivo in termini fisici o meccanici.

Tuttavia le funzioni ricorsive non convogliano naturalmente la nozione di calcolabilità. Turing, nel '36, introdusse un modello di calcolo che si basa sul processo di calcolo mentale e umano: la macchina di Turing o, come Turing la chiamava, la LCM (Logical Computing Machine). Definiamo in modo succinto il concetto di Macchina di Turing[3].

Fissiamo un simbolo che chiamiamo *blank* e che denotiamo con $\Box$.

Una *Macchina di Turing* MT è specificata da tre insiemi finiti Γ, Q e R, come segue:

- Γ è l'*alfabeto*, $\Box \notin \Gamma$;
- Q è l'insieme degli *stati*;
- R è l'insieme delle *istruzioni*. Ciascuna istruzione è una quintupla del tipo $\langle q, s, q', s', \delta \rangle$, dove $q, q' \in Q$, $s, s' \in \Gamma \cup \{\Box\}$ e $\delta \in \{-, 0, +\}$.

[3]Ci sono formulazioni della Macchina di Turing matematicamente molto più sofisticate di quella fornita qui. Per una definizione moderna si veda, per esempio, Papadimitriou [33].

Indichiamo con α una sequenza finita, o stringa, di simboli di Γ e indichiamo con Γ^* l'insieme di tali stringhe, dove $*$ è l'operatore stella di Kleene.

Alla descrizione formale data sopra è in genere associata una descrizione fisica che si basa sulle seguenti componenti: un nastro infinito, costituito di celle, ciascuna cella può contenere un simbolo di Γ oppure $\Box$; una testina di lettura/scrittura che si muove a destra o a sinistra della cella in cui si trova; un'unità di controllo che contiene l'insieme delle istruzioni R e che si trova a ogni passo in uno specifico stato di Q. Il contenuto di una sequenza di celle contigue è una stringa di simboli di $\Gamma \cup \{\Box\}$, cioè un elemento di $(\Gamma \cup \{\Box\})^*$.

Una *configurazione* c_i è una sequenza del tipo $\alpha\ q\ \beta$, dove $q \in Q$ e $\alpha, \beta \in (\Gamma \cup \{\Box\})^*$. Ad esempio, supponiamo che $\Gamma = \{1, 0\}$, allora $11q01\Box1 \in (\Gamma \cup \{\Box\})^*$ è la configurazione in cui la testina si trova sul simbolo 0 e il contenuto del nastro è $1101\Box1$. Notiamo che i blank a sinistra e a destra della stringa $11q01\Box1$ sono omessi.

Una *transizione* consiste nel passaggio da una configurazione c_i a una configurazione c_j. Se nello stato q la testina si trova sul simbolo $s \in \Gamma \cup \{\Box\}$ allora $\langle q, s, q', s', \delta\rangle$ ha il seguente significato: lo stato da q diventa q', la testina scrive s' al posto di s e poi si sposta a destra se $\delta = +$, a sinistra se $\delta = -$, non fa null'altro se $\delta = 0$. Osserviamo che per cancellare un simbolo da una cella è sufficiente scrivere $\Box$, ovvero avremo un'istruzione del tipo $\langle q, s, q', \Box, 0\rangle$.

Una *configurazione di arresto* è una configurazione in cui nessuna istruzione si può applicare.

Una *computazione* o calcolo è un insieme di transizioni. Una computazione può essere infinita oppure finita e terminante in una configurazione di arresto. Possiamo indicare una computazione di lunghezza n nel seguente modo:

$$c_0 \to c_1 \to \cdots \to c_n$$

Una macchina di Turing è utilizzata in due modi: come trasduttore, oppure come riconoscitore per un linguaggio. La prima funzionalità è quella che permette di definire le funzioni Turing-calcolabili e viene analizzata qui di seguito; la seconda verrà analizzata nel prossimo paragrafo.

La definizione di computazione data sopra lascia libera la scelta della istruzione da applicare a una configurazione data.

Una *macchina di Turing deterministica* (MTD) è una macchina $\langle \Gamma, Q, R\rangle$ tale che per ciascun $\langle q, s\rangle \in Q \times \Gamma \cup \{\Box\}$ esiste al più una istruzione $\langle q, s, q', s', \delta\rangle \in R$. Supponiamo inoltre che sia designato uno stato iniziale q_0, per cui una configurazione iniziale sarà della forma $q_0\alpha$, con $\alpha \in \Gamma^*$. Data un MTD, per ciascuna configurazione iniziale esiste un'unica computazione.

Sia $f : \Gamma^* \mapsto \Gamma^*$. Una MTD $\langle \Gamma, Q, R\rangle$ *calcola* f se per ciascun $\alpha \in \Gamma^*$, la computazione iniziata nella configurazione iniziale $q_0\alpha$ termina in una configurazione βq e $f(\alpha) = \beta$.

Definizione 1.10 *Una funzione f è* Turing-calcolabile *o* T-calcolabile *se esiste una MTD che la calcola.*

Per esempio, la funzione successore è T-calcolabile. Essa è calcolata dalla seguente Macchina di Turing:

$$\Gamma = \{1\}$$
$$Q = \{q_0, q_1\}$$
$$R = \{\langle q_0, 1, q_0, 1, + \rangle, \langle q_0, \Box, q_1, 1, + \rangle\}$$

L'alfabeto Γ consiste di un solo simbolo, la cifra 1. Esso è adeguato, in quanto i numeri naturali si possono codificare con sequenze di $n + 1$ cifre 1. Ovvero 0 è codificato da 1, 1 è codificato da 11, eccetera. Q contiene solo due stati: lo stato iniziale q_0, in cui la testina si trova sull'ultima cifra del numero naturale codificato, oppure su un blank; lo stato finale, q_1, in cui la testina si è spostata alla destra dell'ultima cifra e vi ha scritto un 1. Diciamo che q_1 è uno stato finale perché non vi è nessuna istruzione applicabile in q_1.

L' istruzione $\langle q_0, 1, q_0, 1, + \rangle$ dice che se la testina sta su un 1 (che potrebbe essere la codifica del numero 0, oppure l'ultima cifra della codifica del numero di cui si vuole calcolare il successore), allora scrive 1 e si sposta di una cella a destra. La seconda istruzione $\langle q_0, \Box, q_1, 1, 0 \rangle$ dice che se la testina si trova su un blank, deve essere scritto un 1 al posto del blank e $\delta = 0$ indica che la testina non deve fare altro. Osserviamo, in particolare, che la macchina descritta è *deterministica* perché per ciascuna coppia $\langle q, s \rangle$ esiste un'unica istruzione in R.

Analogamente a Church, sempre nel 1936, Turing postula la seguente ipotesi di lavoro:

> [TESI DI TURING, 1936]
> Una Macchina di Turing può eseguire qualunque calcolo che può essere descritto come puramente meccanico.

La tesi di Turing dice che, se esiste una procedura effettiva per ottenere il valore di una funzione matematica, allora tale funzione è T-calcolabile. Come abbiamo visto, la nozione di funzione calcolata da una macchina di Turing è una nozione formale come quella di funzione ricorsiva. Le tesi di Church e Turing stabiliscono dunque una corrispondenza tra due nozioni formali – funzioni ricorsive e funzioni T-calcolabili – e tra esse e la nozione informale di procedura effettiva. Kleene mostra l'equivalenza delle tesi di Church e Turing, e quindi possiamo riferirci a esse come alla *tesi di Church-Turing*. In effetti a seguito della tesi di Church-Turing, un importante risultato ottenuto nella prima metà del secolo scorso (il '900) è che, diversi modelli di calcolo come Macchine di Turing, lambda-calcolo, funzioni ricorsive, sistemi di Post, e altri, sono equivalenti, nel senso che calcolano tutti lo stesso insieme di funzioni. Questi teoremi di equivalenza, insieme alla tesi di Church-Turing, permettono di asserire che una funzione è calcolabile se può essere calcolata in uno qualunque dei

sistemi suddetti, e cioè *l'insieme di tutte le funzioni calcolabili* coincide con l'insieme delle funzioni calcolate in uno qualunque di tali formalismi.

La tesi di Church-Turing e i risultati di equivalenza summenzionati hanno delle conseguenze pratiche importanti:

1. Non scrivere un programma per una funzione che si è dimostrato che non è ricorsiva.
2. Se si è verificato che una funzione è calcolabile, allora essa ha un algoritmo esprimibile in uno qualunque dei formalismi, tra quelli equivalenti visti sopra.
3. Se si è verificato che una funzione è calcolabile in un dato formalismo, lo è anche in un altro – tra quelli equivalenti – anche se non si sa come esprimerla.
4. Si può parlare genericamente di algoritmo che calcola una funzione senza bisogno di specificare il modello di calcolo utilizzato.

Esempio 15 *La funzione $f(x) =$ la x-esima cifra nella espansione decimale di π è calcolabile.*

È facile verificare che la funzione f è ricorsiva. Una procedura effettiva potrebbe essere quella che calcola la prima cifra dell'espansione decimale di π, poi la seconda, la terza, eccetera, e va avanti fino alla x-esima cifra.

Esempio 16 *La funzione $g : \mathbb{N}^n \mapsto \mathbb{N}$ definita come*

$$g(a_1, \ldots, a_n) = 2^{a_1} \times 3^{a_2} \times \ldots \times p_n^{a_n}$$

dove p_n è l'ennesimo numero primo, è calcolabile.

La g è ricorsiva; una procedura effettiva per calcolarla richiede la composizione di funzioni: una funzione per calcolare l'n-esimo numero primo, una funzione per calcolare la potenza $p_k^{a_k}$ e una funzione per calcolare il prodotto dei termini trovati con le precedenti due funzioni. Ad esempio,

$$g(3, 1, 4, 6) = 2^3 \times 3^1 \times 5^4 \times 7^6.$$

Si vedano gli esercizi 24 e 25.

Facciamo ora degli esempi di funzioni che si sa essere calcolabili, ma per le quali non si conosce un algoritmo.

Esempio 17 *La funzione $h(x) = 1$ se esiste una sequenza di almeno x cifre consecutive uguali a 5 nella espansione decimale di π, $h(x) = 0$ altrimenti, è calcolabile.*

La h è calcolabile, anche se non conosciamo un algoritmo che la calcola. Infatti h è ricorsiva: o essa è la funzione costante $h(x) = 1$ h oppure deve esistere un k tale che:

$$h(x) = \begin{cases} 1 & \text{per } x \leq k \\ 0 & \text{per } k < x. \end{cases}$$

In entrambi i casi essa è ricorsiva; quindi sappiamo che esiste un algoritmo, ma non si conosce l'algoritmo che calcola la h perché non si conosce il valore di k.

D'altro canto, dall'argomento presentato precedentemente sull'insieme delle funzioni dai naturali ai naturali e dalla tesi di Church-Turing si evince che esistono funzioni, dai naturali ai naturali, che non sono calcolabili.

Esempio 18 *La funzione $h'(x) = 1$ se esiste una sequenza di esattamente x cifre consecutive uguali a 5 nella espansione decimale di π, $h'(x) = 0$ altrimenti, non è calcolabile.*

Per la h' si veda l'esercizio 29.

1.7.1 Esercizi

Esercizio 21 *Si consideri una macchina dotata di un nastro infinito su cui sono scritte tutte le addizioni tra numeri naturali, cioè $1+0=1, 1+1=2, 1+2=3, \ldots$. Si consideri la seguente procedura:*

`Leggi due numeri naturali` n `ed` m`;`
`Guarda se` $n+m$ `oppure` $m+n$ `è scritto sul nastro;`
`Stampa il risultato.`

La procedura sopra definita è effettiva? Si forniscano accurate motivazioni per la risposta.

Esercizio 22 *Definire induttivamente l'insieme di istruzioni (di un sottoinsieme) del Pascal, dato un insieme iniziale W di simboli.*

Esercizio 23 *Dimostrare che la funzione $f : \mathbb{N} \mapsto \mathbb{N}$ definita come: $f(x) = 1$ se x è un numero primo, 0 altrimenti, è calcolabile.*

Esercizio 24 *Considerare la funzione g definita nell'esempio 16; dire se g^-1 è calcolabile. Se è calcolabile calcolare $g^{-1}(180)$.*

Esercizio 25 *Dimostrare che se $f(x)$ e $g(x)$ sono calcolabili allora la funzione $h(x) = g(f(x))$ è calcolabile.*

Esercizio 26 *Implementare in un linguaggio di programmazione a piacere un programma che stampa ogni numero pari minore uguale a mille come somma di due primi.*

Esercizio 27 *Dimostrare che le seguenti funzioni sono ricorsive:*

1. *Il predecessore p: $(p(0) = 0; \quad p(x+1) = x)$.*

2. *Le relazioni* $\leq$ *e* $=$*;*
3. *La distanza* $|x - y|$*.*

Esercizio 28 *Definire una funzione che non è calcolabile.*

Esercizio 29 *Spiegare perchè la* h' *nell'esempio 18 non è calcolabile; dire se in questo caso sia possibile stabilire che esiste un* k*, come nell'esempio 17.*

1.8 Problemi di decisione

Per introdurre i problemi di decisione diamo una definizione di *problema* diversa da quella usuale in matematica. Un problema $\mathcal{P}$ è costituito da:

1. un insieme di istanze;
2. una richiesta che può avere come risposta *sì* oppure *no*.

Ciascuna istanza è un oggetto finito che deve essere rappresentato effettivamente e uniformemente da una stringa su un alfabeto finito. In questo paragrafo supporremmo sempre che una istanza sia codificata da un numero naturale.

La rappresentazione finita, effettiva e uniforme delle istanze di un problema è cruciale. Supponiamo di volere rappresentare come istanze di un problema un insieme infinito di coppie $\langle n, \sqrt{n}\rangle$, con n un intero. $\sqrt{n}$ potrebbe essere un numero irrazionale, o un numero complesso; quindi a seconda del valore di n la sua rappresentazione sarebbe di volta in volta diversa ed eventualmente infinita. La rappresentazione finita, uniforme ed effettiva di $\sqrt{n}$ significa che può essere definita una codifica di $\sqrt{n}$ tramite una prefissata sequenza di interi. Per un esempio di codifica di una sequenza di interi si vedano l'esempio 16 e l'esercizio 24.

Esempio 19 *Si considerino i seguenti due problemi:*

Problema A
Istanza: *Un numero rappresentato come il prodotto dei suoi divisori.*
Richiesta: *Questo numero è primo?*

Problema B
Istanza: *Una sequenza finita di coppie di stringhe:* $\langle u_1, v_1\rangle\langle u_2, v_2\rangle \cdots \langle u_n, v_n\rangle$*;*
Richiesta: *Esiste una sequenza di indici* $i_1 \cdots i_k$ *tale che,* $u_1 \cdots u_k = v_1 \cdots v_k$*?*

Dato un problema $\mathcal{P}$, il *problema complementare*, che denotiamo con $C\mathcal{P}$ è il problema ottenuto da $\mathcal{P}$ mantenendo la stessa rappresentazione dei dati e negando l'enunciato nella richiesta.

Esempio 20 *Consideriamo il problema A dell'esempio 19 e poniamo come richiesta "Questo numero è diverso da un numero primo?".*

A ciascun problema, costituito da un insieme finito o numerabile di istanze e da una richiesta è associato un linguaggio, detto *linguaggio caratteristico* del problema, che indichiamo con $\mathcal{L}_\mathcal{P}$. Il linguaggio è formato da tutte le istanze del problema per le quali la risposta alla richiesta è *sì*. Il *linguaggio complementare*, indicato con $C\mathcal{L}_\mathcal{P}$ è il linguaggio del problema complementare.

Esempio 21 *Il linguaggio $\mathcal{L}_{\mathcal{P}_A}$ del problema A dell'esempio 19 è*

$$\{1 \times 2, 1 \times 3, 1 \times 5, \ldots\}$$

Il linguaggio complementare $C\mathcal{L}_\mathcal{P}$:

$$\{1, 1 \times 2 \times 2, 1 \times 2 \times 3, 1 \times 2 \times 4, \ldots\}.$$

Un *problema di decisione* consiste nel determinare se una certa istanza del problema $\mathcal{P}$ appartiene a $\mathcal{L}_\mathcal{P}$ oppure a $C\mathcal{L}_\mathcal{P}$. Quindi un algoritmo di decisione equivale al calcolo della funzione caratteristica di$\mathcal{L}_\mathcal{P}$ e cioè $1_{\mathcal{L}_\mathcal{P}} : \mathbb{N} \mapsto \{0,1\}$[4]:

$$1_{\mathcal{L}_\mathcal{P}}(x) = \begin{cases} 1, & \text{se } x \in \mathcal{L}_\mathcal{P} \\ 0, & \text{altrimenti} \end{cases}$$

Per potere determinare la natura di $\mathcal{P}$ è quindi necessario determinare la natura di $\mathcal{L}_\mathcal{P}$. Introduciamo a tal fine la seguente definizione.

Definizione 1.11 [INSIEMI RICORSIVI E RICORSIVAMENTE ENUMERABILI] *Sia X un insieme con elementi codificati in $\mathbb{N}$.*

1. *X è* ricorsivo *se la sua funzione caratteristica $1_X : \mathbb{N} \mapsto \{0,1\}$ è ricorsiva.*
2. *X è* ricorsivamente enumerabile *se è vuoto oppure è l'immagine di una funzione ricorsiva.*

Abbiamo i seguenti risultati:

Teorema 1.12 *Un insieme X è ricorsivo sse entrambi X e il suo complemento sono ricorsivamente enumerabili.*

Dimostrazione Lasciata al lettore (si veda l'esercizio 34). □

Teorema 1.13 *Se $\mathcal{R}$ è la classe degli insiemi ricorsivi abbiamo che se $A, B \in \mathcal{R}$ allora $A \cup B \in \mathcal{R}$, $A \cap B \in \mathcal{R}$, $\overline{A} \in \mathcal{R}$, $A \backslash B \in \mathcal{R}$.*

Dimostrazione Lasciata al lettore (si veda l'esercizio 35). □

[4]Ricordiamo che qui abbiamo assunto che una istanza sia codificata da un numero naturale.

Definizione 1.14 [PROBLEMI DECIDIBILI E SEMIDECIDIBILI] *Sia X un insieme con elementi codificati in $\mathbb{N}$. Il problema $x \in X$?*

1. *È* decidibile *se X è ricorsivo.*
2. *È* semi-decidibile *se X è ricorsivamente enumerabile.*

Poiché ogni funzione ricorsiva è T-calcolabile possiamo riformulare i concetti definiti precedentemente nel seguente modo:

Definizione 1.15 *Un linguaggio $\mathcal{L}_\mathcal{P}$ è* decidibile *(o* Turing-decidibile*) se la funzione caratteristica $1_{\mathcal{L}_\mathcal{P}}$ è calcolabile da una macchina di Turing.*

Quindi, se M è la macchina di Turing che calcola $1_{\mathcal{L}_\mathcal{P}}$, diciamo che M *decide* $\mathcal{L}_\mathcal{P}$ o che M è una *procedura di decisione* per $\mathcal{L}_\mathcal{P}$.

Abbiamo visto nel paragrafo precedente che una macchina di Turing può essere utilizzata non solo per calcolare una funzione, ma anche per riconoscere, o accettare, un linguaggio. Spieghiamo, informalmente, cosa questo significa.

Una *macchina di Turing con stati di accettazione* (MTA), è una quadrupla $\langle \Gamma, Q, R, A \rangle$, dove A è un sottoinsieme di Q detto insieme degli stati di accettazione. Una stringa x è accettata da una MTA se esiste una computazione che, partendo da una configurazione iniziale con x sul nastro, si arresta in uno stato di accettazione. Nel caso in cui la macchina di Turing con stati di accettazione sia deterministica abbiamo i seguenti casi:

1. il calcolo non termina;
2. il calcolo si arresta in uno stato che non è di accettazione;
3. il calcolo si arresta in uno stato di accettazione.

Definizione 1.16 *Il linguaggio $\mathcal{L}$ è accettato da una macchina di Turing deterministica con stati di accettazione MTDA se $\mathcal{L}$ è l'insieme delle stringhe accettate da MTDA.*

Quindi diciamo che un linguaggio è *accettabile* (o *Turing-accettabile*) se esiste una MTDA che lo accetta.

Possiamo riformulare i concetti espressi nella definizione 1.11 come segue:

Definizione 1.17

1. *$\mathcal{L}$ è* ricorsivo *se esiste una MTDA che lo accetta e si arresta su tutti i dati.*
2. *Un linguaggio $\mathcal{L}$ è* ricorsivamente enumerabile *se esiste una MTDA che lo accetta.*

Osserviamo, dalla definizione precedente, che $\mathcal{L}$ è ricorsivo se per ogni elemento di $\mathcal{L}$ si possono verificare solo le due seguenti situazioni:

1. il calcolo si arresta in uno stato di accettazione;
2. il calcolo si arresta in uno stato di non accettazione.

Dalla definizione segue anche che possono esserci linguaggi accettabili che non sono decidibili. Inoltre, la nozione di macchina di Turing deterministica con stati di accettazione non è restrittiva, infatti valgono i seguenti teoremi:

Teorema 1.18 *Un linguaggio è accettato da una macchina di Turing deterministica con stati di accettazione sse esiste una macchina di Turing (non deterministica) con stati di accettazione che lo accetta.*

Omettiamo le dimostrazioni di questo teorema e dei successivi perchè necessiterebbero di uno studio più approfondito delle macchine di Turing e delle sue varianti. Per una dimostrazione dei teoremi 1.18, 1.19, e 1.20, si veda Papadimitriou [33].

Teorema 1.19 *Un linguaggio è accettato da una macchina di Turing con stati di accettazione sse esiste una macchina di Turing (non deterministica) che lo accetta.*

In effetti la nozione di arresto in uno stato di accettazione ha un equivalente più generale per cui se una macchina di Turing con stati di accettazione accetta un linguaggio $\mathcal{L}$ allora esiste una macchina di Turing che *riconosce* ogni stringa di $\mathcal{L}$.

Un modo alternativo di accettare un linguaggio, per una macchina di Turing, consiste nel generare i suoi elementi. Vale il seguente teorema:

Teorema 1.20 *Sia $\mathcal{L}$ un linguaggio. $\mathcal{L}$ è Turing-accettabile sse esiste una macchina di Turing che lo enumera.*

È interessante notare che se $\mathcal{L}$ è ricorsivamente enumerabile allora è numerabile. Il viceversa non vale: un linguaggio numerabile non è necessariamente ricorsivamente enumerabile. Abbiamo infatti il seguente teorema:

Teorema 1.21

1. *L'insieme delle funzioni ricorsive è numerabile.*
2. *L'insieme delle funzioni ricorsive non può essere enumerato da una funzione ricorsiva.*

Dimostrazione Per il primo punto abbiamo visto che l'insieme delle macchine di Turing è numerabile e quindi dalla tesi di Church-Turing consegue che l'insieme delle funzioni calcolabili e quindi ricorsive è numerabile.

Per il secondo punto, sia $\{f_n\}_{n\in\mathbb{N}}$, una enumerazione delle funzioni ricorsive. Supponiamo che l'insieme delle funzioni ricorsive possa essere enumerato da una funzione ricorsiva $F(n,m)$ che sarà $f_n(m)$ nella enumerazione. Se $f_n(m)$ fosse ricorsiva allora la funzione $g(n) = F(n,n) + 1$ sarebbe anche essa ricorsiva. Infatti possiamo trovare un semplice algoritmo che calcola $F(n,n)$, che per ipotesi è ricorsiva, e ci aggiungiamo 1. Quindi per la tesi di Church-Turing abbiamo che g è

calcolabile e quindi ricorsiva, e quindi $g = f_k$, per un f_k nella lista delle funzioni ricorsive. Ma allora $g(k) = f_k(k) = F(k,k) = g(k) - 1$; contraddizione. □

Il precedente teorema ci dice che, nonostante le funzioni ricorsive siano in quantità numerabile, non esiste una MT che le enumera. Questo problema è analogo al problema della fermata che vedremo più sotto. Quindi la nozione di insieme enumerabile da una macchina di Turing è più forte di quella di insieme numerabile.

Ricorrendo alla tesi di Church-Turing possiamo dire che se esiste una *procedura effettiva* che enumera $\mathcal{L}$ allora $\mathcal{L}$ è ricorsivamente enumerabile.

Osserviamo che nella nozione di *decidibilità* è implicita la tesi di Church-Turing: decidibile significa che può essere deciso con l'ausilio di un calcolo meccanico, ovvero la funzione caratteristica è calcolabile con l'ausilio di una procedura effettiva.

Ovviamente, se un insieme è finito esso è effettivamente enumerabile, mentre non tutti gli insiemi infiniti lo sono. Il fatto che l'appartenenza o meno di un elemento a un insieme infinito possa essere decisa mediante un algoritmo significa che l'insieme infinito è descrivibile in modo finito (mediante l'algoritmo che ne caratterizza l'appartenenza).

Teorema 1.22 *Esistono linguaggi che non sono ricorsivamente enumerabili.*

Dimostrazione Lasciata al lettore (si veda l'esercizio 36). □

Dal precedente teorema segue che se X non è ricorsivamente enumerabile allora il problema $x \in X?$ non è decidibile e non è neanche semi-decidibile. Comunque, in generale diciamo che se la funzione caratteristica di un insieme non è calcolabile allora il problema si dice *insolubile* o *indecidibile*. La scoperta dell'esistenza di problemi indecidibili è uno dei risultati più significativi della teoria della calcolabilità.

Un esempio di problema insolubile o indecidibile è il *problema della fermata*. Ne diamo una descrizione informale, che fa uso della tesi di Church-Turing.

Si consideri il modello di calcolo della macchina di Turing, abbiamo visto che se f è una funzione calcolabile allora esiste una macchina di Turing che la calcola. Abbiamo visto inoltre che l'insieme degli algoritmi è numerabile, quindi possiamo supporre che sia possibile codificare ciascuna macchina con un determinato numero x. Ovvero x è il numero della x-esima macchina di Turing, che indichiamo con P_x, che calcola la x-esima funzione f. Inoltre, questa codifica è biunivoca, ovvero, a ogni numero naturale corrisponde un unica macchina di Turing e a ogni macchina di Turing è associato un unico numero naturale. Ci si convinca che ciò è possibile. Ci si convinca, inoltre, che tale codifica ci permette di trattare un numero naturale e una macchina – ovvero un insieme di istruzioni o programma – allo stesso modo. Così un programma – un insieme di istruzioni – che prende come input un numero naturale, prende come input anche il programma associato a quel numero naturale. Il problema della fermata si può quindi esprimere come segue:

[Problema della Fermata]
Siano x, y una coppia di interi. Sia P_x la macchina di Turing codificata

da x. P_x si arresta sul dato y?

Mostriamo, informalmente, che il problema descritto, non è decidibile. Se lo fosse, per quanto asserito sopra, potremmo definire una funzione ricorsiva g del tipo:

$$g(x,y) = \begin{cases} 1 \text{ se } P_x \text{ si arresta sul dato } y \\ 0 \text{ altrimenti} \end{cases} \tag{1.1}$$

Siccome abbiamo assunto g ricorsiva, possiamo definire la seguente funzione:

$$\psi(x) = \begin{cases} 1 \text{ se } g(x,x) = 0 \\ \text{indefinita se } g(x,x) = 1 \end{cases} \tag{1.2}$$

La ψ è calcolabile perchè è definita tramite la g e siccome $g(x,x)$ deve essere uguale a 1 oppure uguale a 0 ciò ci fornisce un algoritmo per calcolare la ψ. Per la tesi di Church-Turing possiamo concludere che esiste una macchina di Turing P_z che calcola la ψ. Per l'ipotesi di codifica biunivoca, esiste un numero naturale z associato a P_z. Abbiamo per (1.1):

$$g(z,z) = 1 \text{ sse } P_z \text{ si arresta su } z$$

Ma abbiamo anche per (1.2):

$$\psi(z) = 1 \text{ sse } g(z,z) = 0$$

e siccome P_z è la macchina di Turing che calcola ψ, e z è la codifica di P_z, abbiamo

$$g(z,z) = 1 \text{ sse } g(z,z) = 0$$

contraddizione. Perciò g non è calcolabile per la tesi di Church-Turing e quindi il problema della fermata non è risolvibile, ovvero non è decidibile.

In realtà i problemi indecidibili sono assai numerosi, come dimostra il teorema che enunciamo più sotto. Ricordiamo prima che una proprietà di un elemento di un insieme X si dice *banale* se tutti gli elementi di X hanno tale proprietà o non ce l'ha nessuno.

Teorema 1.23 [RICE] *Tutte le proprietà* non banali *dei linguaggi ricorsivamente enumerabili sono indecidibili.*

Il teorema di Rice ci dice che se P è una proprietà non banale il seguente problema è indecidibile:

Istanza:	Una macchina di Turing che riconosce un linguaggio ricorsivamente enumerabile $\mathcal{L}$
Richiesta:	$\mathcal{L}$ ha la proprietà P

Per una lista di problemi indecidibili si veda Davis [16]. Ricordiamo, in particolare, il problema della *Corrispondenza di Post* (si veda il problema B dell'esempio 19).

1.8.1 Esercizi

Esercizio 30 *Dimostrare che l'insieme dei numeri pari è decidibile.*

Esercizio 31 *Dire se è vero o falso che l'insieme di tutte le stringhe di lunghezza finita (ma arbitraria) di simboli su un alfabeto finito è decidibile.*

Esercizio 32 *Dimostrare che se un insieme è co-finito, cioè se ha il complemento finito, allora è decidibile.*

Esercizio 33 *Dire se è vero o falso che l'insieme dei programmi PASCAL sintatticamente corretti è decidibile.*

Esercizio 34 *Dimostrare il teorema 1.12: un insieme X è ricorsivo se entrambi X e il suo complemento sono ricorsivi.*

Esercizio 35 *Dimostrare il teorema 1.13. Si usi la funzione caratteristica.*

Esercizio 36 *Dimostrare il teorema 1.22. Si usi la definizione di insieme ricorsivo.*

Esercizio 37 *Si usi il teorema di Rice per dimostrare che le seguenti proprietà di un insieme X ricorsivamente enumerabile non sono decidibili:*

1. *X è ricorsivo;*
2. *X non è ricorsivo;*
3. *$X = \mathbb{N}$.*

1.9 Complessità computazionale

La nozione di calcolabilità si è sviluppata ben prima dell'informatica e tuttavia sono proprio i concetti fondamentali sulla calcolabilità, introdotti nei due precedenti paragrafi, che hanno stimolato e motivato la nascita dello studio della complessità computazionale.

Nei precedenti paragrafi abbiamo visto che esistono dei problemi di decisione che non possono essere risolti da algoritmi, abbiamo in particolare fornito l'esempio del problema della fermata. Abbiamo anche visto, grazie al teorema di Rice, che vi sono numerosi problemi indecidibili. Possiamo pertanto classificare i problemi in due classi, quelli risolvibili tramite algoritmi e quelli non risolvibili.

Se consideriamo i problemi risolvibili ci interessa determinare una misura dinamica dell'andamento della computazione, ovvero una misura dei passi elementari

necessari alla risoluzione del problema. Chiaramente la misura dipende dal modello di calcolo.

La *teoria astratta della complessità*, dovuta a Manuel Blum, fornisce due assiomi che definiscono il concetto di misura dinamica di una computazione, indipendentemente dal modello di calcolo scelto.

In questo paragrafo, per introdurre i concetti basilari della complessità computazionale, ci riferiremo al modello di calcolo della macchina di Turing.

Abbiamo visto, nel paragrafo 1.7, che una computazione di m passi

$$c_0 \to c_1 \to \cdots \to c_m$$

ha lunghezza m.

Definizione 1.24 *Sia $\langle \Gamma, Q, R \rangle$ una macchina di Turing deterministica MTD e siano S e T funzioni da $\mathbb{N}$ in $\mathbb{N}$:*

1. *diciamo che $\langle \Gamma, Q, R \rangle$ ha* complessità temporale *(inferiore a) T se la lunghezza del calcolo di tutte le stringhe di lunghezza n di Γ^* è al più $T(n)$;*
2. *diciamo che $\langle \Gamma, Q, R \rangle$ ha* complessità spaziale *(inferiore a) S se nel corso del calcolo di ciascuna stringa di lunghezza n di Γ^* al più $S(n)$ celle sono state usate dalla testina della MTD.*

Definizione 1.25 *Sia $\langle \Gamma, Q, R \rangle$ una macchina di Turing deterministica MTD:*

1. *sia $\mathcal{L}$ un sottoinsieme di Γ^*. Il problema $x \in \mathcal{L}$? ha complessità T se esiste una MTD che ha complessità T che accetta $\mathcal{L}$;*
2. *sia $f : \Gamma^* \mapsto \Gamma^*$; f ha complessità T se esiste una MTD di complessità T che calcola f.*

Una *classe* $\mathcal{C}$ è un insieme di linguaggi costruiti su un determinato alfabeto. Data la corrispondenza fra problemi e linguaggi, una classe di linguaggi denota una classe di problemi. Il complemento di una classe $\mathcal{C}$, denotato con *co*$\mathcal{C}$ è l'insieme dei linguaggi complementari ai linguaggi di $\mathcal{C}$. Un linguaggio $\mathcal{L}$ si dice *completo* rispetto a una classe $\mathcal{C}$ se qualunque altro linguaggio $\mathcal{L}'$ della classe può essere ridotto a $\mathcal{L}$. Per *riduzione* intendiamo che esiste una funzione f, totale (si veda la definizione a pagina 8) e calcolabile tale che per ogni stringa x dell'alfabeto di $\mathcal{L}'$ abbiamo[5]:

$$x \in \mathcal{L}' \text{ sse } f(x) \in \mathcal{L}.$$

Per giustificare quanto sopra abbiamo assunto che le istanze di un problema siano opportunamente codificate in modo effettivo e uniforme. In particolare, in questi paragrafi abbiamo sempre assunto che le istanze di un problema siano codificate da numeri naturali.

[5] Possono essere essere utilizzate anche altre nozioni di riducibilità.

Introduciamo la notazione $O(f(x))$; essa si riferisce a una classe di funzioni che possono essere maggiorate, a meno di costanti, da parte della funzione f. Se f è una funzione da $\mathbb{N}$ in $\mathbb{N}$:

$$O(f) = \{g : \mathbb{N} \mapsto \mathbb{N} \mid \quad \exists c_1, c_2 \in \mathbb{R} \quad \exists n_0 \in \mathbb{N} \quad \forall n \geq n_0 (g(n) \leq c_1 + c_2 \ f(n)\}$$

Scriviamo $g(n) = O(f(n))$ piuttosto che $g(n) \in O(f(n))$.

Riprendendo le definizioni 1.24 e 1.25, possiamo denotare con $\mathcal{L}_t$ il linguaggio $\mathcal{L}$ riconosciuto da una MTD che ha complessità t. Diamo ora la seguente definizione

Definizione 1.26 *Una classe di complessità deterministica rispetto al tempo $\mathcal{C}_t$ è definita come:*

$$\mathcal{C}_t = \{\mathcal{L}_t \ | t(x) = O(T(x))\}.$$

La definizione 1.26 dice che una classe $\mathcal{C}_t$ consiste di tutti quei linguaggi accettati da macchine di Turing che hanno complessità t, dove t appartiene alla classe delle funzioni che possono essere maggiorate da T, ovvero che hanno T come funzione limite.

Introduciamo ora la classe $\mathcal{P}$ che include quei problemi che sono considerati semplici o *trattabili*. Sia $p_k(n) = p^k$ e sia $\mathcal{C}_{p_k}$ la corrispondente classe, come da definizione 1.26. La classe $\mathcal{P}$ è esprimibile come:

$$\mathcal{P} = \mathcal{C}_{p_0} \cup \mathcal{C}_{p_1} \ldots \cup \mathcal{C}_{p_k} \ldots$$

e siccome ogni polinomio di grado k appartiene alla classe di funzioni $O(n^k)$ allora la classe $\mathcal{P}$ coincide con l'unione delle classi di complessità che hanno come funzione limite un polinomio di grado finito.

Informalmente, un algoritmo è polinomiale se richiede un numero $O(n^k)$ di passi per essere risolto. Un problema $x \in A$ è polinomiale se $\mathcal{L}_A \in \mathcal{P}$ o se esiste un algoritmo polinomiale che lo risolve. Una funzione f è una trasformazione polinomiale se esiste un algoritmo polinomiale che la calcola.

La classe $\mathcal{P}$ è chiusa rispetto alla complementazione. Ovvero se $co\mathcal{P} = \{\mathcal{L} | C\mathcal{L} \in \mathcal{P}\}$ abbiamo $co\mathcal{P} = \mathcal{P}$.

[TESI DI COOK]
I problemi trattabili sono quelli della classe $\mathcal{P}$.

Possiamo ora riformulare la definizione 1.24 per macchine di Turing non deterministiche:

Definizione 1.27 *Sia $\mathcal{L} \subseteq \Gamma^*$. Una macchina di Turing non deterministica $\langle \Gamma, Q, R \rangle$ accetta $\mathcal{L}$ in tempo T su un sottoinsieme Q_0 di Q se per ogni $x \in \mathcal{L}$ esiste una computazione di x che termina in uno stato di Q_0 di lunghezza inferiore a $T(|x|)$.*

Possiamo, quindi, scrivere una definizione analoga a 1.26 per classi non deterministiche ovvero per classi che contengono i linguaggi decisi in tempo t da macchine di Turing non deterministiche:

Definizione 1.28 *Una classe non deterministica è definita come:*

$$\mathcal{NC}_t = \{\mathcal{L}_t \mid t(x) = O(T(x))\}.$$

La classe $\mathcal{NP}$ si può definire analogamente alla classe $\mathcal{P}$ prendendo l'unione di tutte le classi non deterministiche limitate superiormente da un polinomio in k:

$$\mathcal{NP} = \mathcal{NC}_{p_0} \cup \mathcal{NC}_{p_1} \ldots \cup \mathcal{NC}_{p_k} \ldots$$

In altre parole, $\mathcal{L}$ è in $\mathcal{NP}$ se esiste una macchina di Turing non deterministica che accetta $\mathcal{L}$ in tempo polinomiale.

Teorema 1.29 *Se $\mathcal{L}$ è accettato da una macchina di Turing in tempo $T(n) = O(n^k)$ allora esiste una costante c e una MTD di complessità $O(c^{n^k})$ che risolve il problema $x \in \mathcal{L}$.*

Dimostrazione Sia d il grado di non determinismo di una macchina di Turing, ovvero il numero massimo di quintuple $\langle q, s \ldots\rangle$ che cominciano con la stessa coppia $\langle q, s\rangle$. Osserviamo che è la scelta tra queste istruzioni che crea il non determinismo, infatti per una MTD $d = 1$. Per $x \in \Gamma^*$ ci sono al più $d^{T(|x|)}$ differenti calcoli di lunghezza $\leq T(|x|)$, per x. A partire da questi è possibile costruire una MTD che accetta $\mathcal{L}$ e si dimostra che $\mathcal{L}$ è in $\mathcal{NP}$. □

La classe $\mathcal{NP}$ è la classe dei problemi per i quali esiste un algoritmo di risoluzione non deterministico di complessità polinomiale. La classe $\mathcal{NP}$ si può definire analogamente alla classe $\mathcal{P} \subseteq \mathcal{NP}$, in quanto un algoritmo deterministico è un caso particolare di un algoritmo non deterministico.

Abbiamo visto precedentemente cosa intendiamo per riduzione di un linguaggio ad un altro linguaggio. Abbiamo dato una definizione di riduzione in termini di una funzione totale calcolabile; abbiamo detto che, se tale funzione è calcolabile in tempo polinomiale la chiamiamo trasformazione polinomiale.

Dati due linguaggi $\mathcal{L}_1$ e $\mathcal{L}_2$ diremo che $\mathcal{L}_1$ è *riducibile in tempo polinomiale* a $\mathcal{L}_2$ se esiste una trasformazione polinomiale t tale che $x \in \mathcal{L}_1$ sse $t(x) \in \mathcal{L}_2$. Indichiamo la riduzione con $\leq_P$.

Un linguaggio si dice $\mathcal{NP}$-*completo* se è nella classe $\mathcal{NP}$ e se *ogni* altro linguaggio della classe è riducibile a esso in tempo polinomiale. Di conseguenza un problema si dice $\mathcal{NP}$-completo se è nella classe $\mathcal{NP}$ ed è completo per riduzione polinomiale.

Proposizione 1.30 *Sia $\equiv_P$ la relazione di equivalenza generata da $\leq_P$:*

1. *Se A e B sono $\mathcal{NP}$-completi allora $A \equiv_P B$;*

2. Se A è $\mathcal{NP}$-completo e B di classe $\mathcal{NP}$ allora B è $\mathcal{NP}$-completo se $A \leq_P B$.

3. Se A è $\mathcal{NP}$-completo e di classe $\mathcal{P}$ allora $\mathcal{P} = \mathcal{NP}$.

Dimostrazione
1. Se A e B sono $\mathcal{NP}$-completi allora $A \leq_P B$ e $B \leq_P A$ e quindi $A \equiv_P B$.
2. Se A è $\mathcal{NP}$-completo e B è di classe $\mathcal{NP}$ allora se $A \leq_P B$ per ogni $\mathcal{L}$ in $\mathcal{NP}$, $\mathcal{L} \leq_P B$ e quindi B è $\mathcal{NP}$-completo.
3. $\mathcal{P} \subseteq \mathcal{NP}$; se A è $\mathcal{NP}$-completo allora per ogni $\mathcal{L}$ in $\mathcal{NP}$, $\mathcal{L} \leq_P A$ e siccome A in $\mathcal{P}$ ne segue che $\mathcal{L}$ in $\mathcal{P}$ e quindi $\mathcal{NP} \subseteq \mathcal{P}$. □

Osserviamo che il primo punto della precedente proposizione ci dice che tutti i problemi $\mathcal{NP}$-completi sono altrettanto difficili. Il secondo ci fornisce il metodo più usuale per dimostrare la $\mathcal{NP}$-completezza di un linguaggio. Il terzo punto ci dice che lo studio dei problemi $\mathcal{NP}$ completi è cruciale al fine di stabilire che $\mathcal{P} \neq \mathcal{NP}$.

Congettura: $\boxed{\mathcal{P} \neq \mathcal{NP}}$

Il primo problema che è stato dimostrato, da Cook, essere $\mathcal{NP}$-completo è la soddisfacibilità di enunciati proposizionali[6]. Sia X un insieme infinito numerabile.

SAT
Istanza: Un enunciato A proposizionale espresso come congiunzione di disgiunzioni di lettere proposizionali p, $p \in X$.
Richiesta: A è soddisfacibile?

Teorema 1.31 [COOK, 1971] *Il problema SAT è $\mathcal{NP}$-completo.*

La dimostrazione è stata fatta nell'articolo [14], in cui Cook ha anche introdotto la nozione di riducibilità polinomiale, la classe $\mathcal{NP}$ e la nozione di problema $\mathcal{NP}$-completo.

Analogamente a $co\mathcal{P}$ definiamo $co\mathcal{NP} = \{\mathcal{L} | C\mathcal{L} \in \mathcal{NP}\}$. Poichè $\mathcal{P}$ è chiuso per complementazione si ha che $\mathcal{P} = co\mathcal{P} \subset \mathcal{NP}$ e quindi $\mathcal{P} = co\mathcal{P} \subset \mathcal{NP} \cap co\mathcal{NP}$. Si congettura che esistano linguaggi che non sono in $\mathcal{P}$ e che sono nell'intersezione $\mathcal{NP} \cap co\mathcal{NP}$. Un altro problema ancora aperto è se la classe $\mathcal{NP}$ è o meno chiusa per complementazione, cioè se dato un linguaggio in $\mathcal{NP}$ il suo complemento è in $\mathcal{NP}$ o meno. Se questo non fosse vero avremmo chiaramente che $\mathcal{P} \neq \mathcal{NP}$, infatti abbiamo visto che $\mathcal{P}$ è chiusa per complementazione. In particolare, non esiste nessun problema $\mathcal{NP}$-completo il cui complemento sia noto essere in $\mathcal{NP}$.

Ricordiamo infine che le tecniche utilizzate per dimostrare la $\mathcal{NP}$-completezza sono utilizzate anche per dimostrare che un problema di decisione sta al di fuori di $\mathcal{NP}$. Tuttavia si dice, in genere, che un problema di decisione, che appartenga a $\mathcal{NP}$ o meno, è $\mathcal{NP}$-hard[7] se esiste una riduzione a un problema $\mathcal{NP}$-completo.

[6] Gli enunciati proposizionali e la forma normale congiuntiva verranno definiti nei capitoli 3 e 4.
[7] $\mathcal{NP}$-hard viene tradotto in italiano come $\mathcal{NP}$-arduo, o $\mathcal{NP}$-difficile.

1.9.1 Esercizi

Esercizio 38 *Definire una macchina di Turing che calcola la somma di due numeri su un alfabeto* $\{1, 0\}$*, quindi numeri binari. Si analizzi la complessità* T *della macchina definita.*

Esercizio 39 *Date due classi* C_1 *e* C_2 *dimostrare che* $C_1 \subseteq C_2$ *sse* $coC_1 \subseteq coC_2$*.*

Esercizio 40 *Dimostrare che non esiste una funzione* t *e una classe* C_t *tale che tutti i linguaggi decidibili sono in* C_t*.*

Esercizio 41 *Dimostrare che se* $L_1, L_2 \in \mathcal{P}$ *allora* $\mathcal{L}_1 \cup \mathcal{L}_2 \in \mathcal{P}$ *e* $\mathcal{L}_1 \cap \mathcal{L}_2 \in \mathcal{P}$*.*

1.10 Riepilogo

In questo capitolo abbiamo introdotto:

1. la nozione di insieme e alcune semplici operazioni su insiemi;
2. la nozione di coppia e di n-upla ordinata;
3. le nozioni di relazione, funzione, operazione e alcune loro proprietà importanti;
4. la cardinalità di insiemi;
5. il metodo di dimostrazione per diagonalizzazione;
6. l'assioma del buon ordinamento;
7. il principio di induzione matematica e quello di induzione completa;
8. la definizione induttiva di insiemi;
9. la ricorsione;
10. le funzioni calcolabili e i modelli di calcolo;
11. i problemi di decisione: le nozioni di decidibilità, semidecidibilità e indecidibilità;
12. abbiamo infine presentato le nozioni di base della complessità computazionale.

Capitolo 2
I sistemi formali

Introduciamo qui la nozione di *sistema formale* o *sistema logico* o, più semplicemente, *logica*.

Definizione 2.1 *Una* logica Λ *è definita da:*

- *Un insieme non vuoto (finito o al più numerabile)* $\mathcal{A}$ *di simboli, detto* alfabeto *di* Λ. *Una sequenza finita di elementi di* $\mathcal{A}$ *è detta* espressione *di* Λ.
- *Un sottoinsieme* $\mathcal{F}$ *delle espressioni di* Λ *chiamato l'insieme delle* formule ben formate *(o più semplicemente* formule) *di* Λ.
- *Un sottoinsieme* Ax *di* $\mathcal{F}$, *detto insieme degli* assiomi logici *o* assiomi *di* Λ.
- *Un insieme finito* $\mathcal{R}$ *di relazioni* $R_1, \ldots, R_n$ *tra formule di* Λ, *dette* regole di inferenza. *Per ciascuna* R_i *di* $\mathcal{R}$ *esiste un unico intero positivo* j *tale che, per ogni insieme costituito da* j *formule e per ogni formula* A, *si può effettivamente decidere se le* j *formule sono in relazione* R_i *con* A. *In caso positivo,* A *è detta* conseguenza diretta *delle formule date, mediante* R_i.

2.1 Il linguaggio

L'insieme $\mathcal{F}$ delle formule ben formate costituisce il *linguaggio* $\mathcal{L}$ della logica. Generalmente $\mathcal{F}$ viene definito in modo induttivo[1]. Diamo qui di seguito una definizione

[1] Oppure attraverso una grammatica libera da contesto. In questo caso è importante che essa non sia ambigua.

induttiva dell'insieme delle formule ben formate $\mathcal{F}$ di una logica Λ in cui distinguiamo un insieme iniziale detto insieme degli atomi $\mathcal{A}tom$ e il cui alfabeto $\mathcal{A}$ contenga un insieme finito di operatori (o connettivi) che, per semplicità, supponiamo unari (li indichiamo con "$\star$" prefisso) e binari (li indichiamo con "$\circ$" infisso).

Definizione 2.2 *L'insieme $\mathcal{F}$ delle* formule ben formate *di una logica Λ con alfabeto $\mathcal{A}$ è l'insieme definito induttivamente come segue:*

1. *Se A è un elemento di $\mathcal{A}tom$ allora $A \in \mathcal{F}$;*
2. *Se $A \in \mathcal{F}$ e $\star$ è un operatore unario allora $(\star A) \in \mathcal{F}$;*
3. *Se $A \in \mathcal{F}$, $B \in \mathcal{F}$ e $\circ$ è un operatore binario allora $(A \circ B) \in \mathcal{F}$.*

La costruzione illustrata è una costruzione induttiva. Come abbiamo visto nel precedente capitolo, ci sono due metodi per costruire tale insieme: uno top-down, ci dice che l'insieme $\mathcal{F}$ è il più piccolo insieme induttivo, cioè l'intersezione di tutti gli insiemi generati dall'insieme iniziale $\mathcal{A}tom$ e chiusi rispetto alle operazioni $\star$ e $\circ$. Un secondo, bottom-up, che ci permette di costruire l'insieme $\mathcal{F}$ a partire dall'insieme iniziale $\mathcal{A}tom$ applicando le operazioni $\star$ e $\circ$ un numero finito di volte. Dal lemma 1.6 e dal principio di induzione ricaviamo che entrambe le costruzioni sono equivalenti e qualunque sottoinsieme S di $\mathcal{F}$, tale che $\mathcal{A}tom \subseteq S$ e S chiuso rispetto a $\star$ e $\circ$, è uguale a $\mathcal{F}$.

Una importante conseguenza del fatto che l'insieme delle formule ben formate sia stato definito induttivamente è che per esso vale un principio di induzione strutturale. Un tale principio si rivela molto utile nel verificare che una data proprietà vale per tutti gli elementi dell'insieme: esso riconduce la verifica al caso base e al caso induttivo della costruzione degli oggetti appartenenti all'insieme.

Teorema 2.3 [PRINCIPIO DI INDUZIONE STRUTTURALE] *Per dimostrare che una proprietà Q vale per ogni formula A di $\mathcal{F}$ basta dimostrare:*

- Passo base
  - *Q vale per ogni formula A di $\mathcal{A}tom$;*
- Passo induttivo
  - *Se la proprietà Q vale per A allora vale per $(\star A)$, per ogni operatore unario $\star$;*
  - *Se la proprietà Q vale per B e C, allora vale per $(B \circ C)$, per ogni operatore binario $\circ$.*

Dimostrazione Sia fbf l'insieme delle formule ben formate di $\mathcal{L}$ che hanno la proprietà Q. Per il passo base e per il passo induttivo, l'insieme soddisfa le condizioni

1-3 della definizione 2.2, pertanto fbf è induttivo. Siccome $\mathcal{F}$ è il più piccolo insieme che contiene $\mathcal{A}tom$ e chiuso rispetto a $\star$ e $\circ$ (ovvero $\mathcal{F}$ è il più piccolo insieme induttivo), allora $\mathcal{F} \subseteq fbf$ e dunque ogni formula di $\mathcal{F}$ ha la proprietà Q. □

È chiaro che non tutte le espressioni di $\mathcal{L}$ sono formule ben formate, tuttavia, data una formula A possiamo considerare A come una *sequenza finita* di simboli di $\mathcal{A}$. Ad esempio se $A = (\star(B \circ (\star(C \circ D))))$, con $B, C, D \in \mathcal{A}tom$, possiamo considerare A come la sequenza di simboli $\langle(,\star,(,B,\circ,(,\star,(,C,\circ,D,),),),)\rangle$. Si chiama *prefisso proprio* di A la stringa non vuota di simboli S tale che A risulta dalla concatenazione di S e R, per qualche stringa non vuota di simboli R. Chiaramente se $A \in \mathcal{A}tom$ A non ha un prefisso proprio.

Osserviamo (si vedano gli esercizi 43 e 44) che il prefisso proprio di una formula ben formata non è esso stesso una formula ben formata.

Il seguente teorema ci dice che l'insieme delle formule ben formate è un insieme liberamente generato.

Teorema 2.4 [UNICITÀ DELLA DECOMPOSIZIONE] *Per ogni formula ben formata A di $\mathcal{F}$ vale una (e una sola) delle seguenti proprietà:*

1. $A \in \mathcal{A}tom$;
2. *Esiste un unico operatore unario $\star$ e un'unica formula B tale che A ha la forma $(\star B)$;*
3. *Esiste un unico operatore binario $\circ$ e due formule uniche B e C tali che A ha la forma $(B \circ C)$.*

Dimostrazione Prima di tutto dobbiamo mostrare che l'insieme delle formule è liberamente generato, cioè che gli operatori sono iniettivi e il loro codominio è a intersezione vuota. Analizziamo solo il caso degli operatori binari.

i. Sia $(B \circ C) = (D \circ E)$; cancelliamo il primo simbolo e otteniamo $B \circ C)$ e $D \circ E)$. Per gli esercizi 43 e 44 abbiamo che $B \circ C) = D \circ E)$ e, siccome B non può essere un prefisso proprio di D, né D può essere un prefisso proprio di B, avremo $B = D$; pertanto $\circ C) = \circ E)$ e quindi, seguendo lo stesso ragionamento, $C = E$.

ii. Dobbiamo ora mostrare che operatori differenti hanno l'intersezione dei loro codomini vuota. Consideriarmo ancora solo il caso di un operatore binario e mostriamo che il suo codominio è a intersezione vuota con quello di ogni altro operatore binario e di un qualunque operatore unario. Supponiamo $\circ \neq \circ'$; dobbiamo mostrare che, se $(B \circ C) = (D \circ' E)$ allora abbiamo una contraddizione. Infatti, per lo stesso ragionamento di prima, se $(B \circ C) = (D \circ' E)$ allora $B = D$ e dunque $\circ = \circ'$; contraddizione. Analogamente, supponiamo che $(B \circ C) = (\star D)$, allora $B \circ C) = \star D)$ e quindi B deve essere della forma $\star E$ e D della forma $\circ C$ ma nessuna formula comincia né con $\star$ – senza parentesi – né con $\circ$.

Abbiamo così mostrato che i connettivi sono iniettivi e i loro codomini sono a intersezione vuota.

Mostriamo, a questo punto, che ogni formula ben formata ricade in almeno una delle categorie enunciate.

Supponiamo che esista una formula ben formata $F \in \mathcal{F}$ che non soddisfa nessuna delle proprietà *1-3*. Sia $X = \mathcal{F} - \{F\}$. Per *1*, $\mathcal{A}tom \subseteq X$. Quindi X è liberamente generato da $\mathcal{A}tom$, per mezzo di $\star$ e $\circ$, dunque è induttivo. Ne segue che X soddisfa le condizioni della definizione 2.2 e, pertanto, $\mathcal{F} \subseteq X$. Per definizione di X, $X \subseteq \mathcal{F}$, ne segue che $X = \mathcal{F}$ Quindi $X = \mathcal{F} = \mathcal{F} - \{F\}$; contraddizione.

Occorre, infine, mostrare che nessuna formula ricade in più di una delle categorie 1-3; questa parte è lasciata al lettore (si veda l'esercizio 45). □

Nel primo caso si dice che A è una formula *atomica*, nel secondo che $\star$ è il suo *operatore principale* e che B è la sua (unica) *sottoformula immediata*, nel terzo caso si dice che $\circ$ è il suo *operatore principale* e che B e C sono le sue (uniche) *sottoformule immediate*.

Il risultato precedente ci permette di definire funzioni sull'insieme $\mathcal{F}$ ricorsivamente.

Teorema 2.5 [PRINCIPIO DI RICORSIONE STRUTTURALE] *Sia f definita per ciascuna formula $A \in \mathcal{A}tom$; esiste una e una sola funzione $\overline{f}$ che estende f tale che:*

> *Il valore di $\overline{f}$ su $(\star A)$ è definito in termini di $\overline{f}(A)$ per ogni operatore unario $\star$;*
>
> *Il valore di $\overline{f}$ su $(A \circ B)$ è definito in termini di $\overline{f}(A)$ ed $\overline{f}(B)$ per ogni connettivo binario $\circ$.*

Dimostrazione È una conseguenza immediata del teorema di ricorsione 1.9 e del precedente teorema 2.4. □

Esempio 22 *Definiamo, ricorsivamente, una funzione che costruisce l'insieme degli atomi che occorrono in una formula di $\mathcal{F}$ come segue:*

$$conta : \mathcal{F} \mapsto 2^{\mathcal{A}tom}$$

1. $conta(A) = \{A\}$ *se* $A \in \mathcal{A}tom$
2. $conta(\star A) = conta(A)$
3. $conta(A \circ B) = conta(A) \cup conta(B)$.

Per il teorema di ricorsione, *conta* esiste ed è unica, ovvero siamo certi che se abbiamo un insieme di k atomi in una formula, non ci sarà una funzione $conta'$, definita nello stesso modo, tale che $conta(A)$ differisce da $conta'(A)$ per qualche A. Sia $A = (\star(B \circ (C \circ D)))$, avremo: $conta(A) = conta(B \circ (C \circ D))) = conta(B) \cup conta(C \circ D) = conta(B) \cup conta(C) \cup conta(D) = \{B, C, D\}$.

Si può mostrare (si veda l'esercizio 43) che ogni formula ben formata non atomica comincia con "(" e termina con ")". In base a tale caratteristica, le parentesi si possono omettere imponendo una precedenza fra i connettivi. Ad esempio, in una logica in cui i connettivi siano: $\neg, \rightarrow, \wedge$ si può convenire che il connettivo $\neg$ leghi più strettamente del connettivo $\wedge$, e che il connettivo $\wedge$ a sua volta leghi più strettamente del connettivo $\rightarrow$. Quando uno stesso connettivo viene utilizzato più volte di seguito si intende che esso associ a destra, cioè $A \circ B \circ C = (A \circ (B \circ C))$.

Esempio 23 *Supponiamo di avere la formula senza parentesi:*

$$\neg A \wedge \neg B \rightarrow C \wedge D \wedge E.$$

La formula è univocamente letta nel seguente modo:

$$(((\neg A) \wedge (\neg B)) \rightarrow (C \wedge (D \wedge E))).$$

2.1.1 Esercizi

Esercizio 42 *Dimostrare che $(A \circ (\circ B))$ non appartiene a $\mathcal{F}$.*

Esercizio 43 *Dimostrare che ogni formula $A \in \mathcal{F}$ ha lo stesso numero di parentesi "(" e di parentesi ")". Usare l'induzione strutturale.*

Esercizio 44 *Provare che ogni prefisso proprio di una formula A ha più parentesi "(" che parentesi ")".*

Esercizio 45 *Dimostrare che per ciascuna formula ben formata vale al più una delle proprietà 1–3 del teorema 2.4.*

Esercizio 46 *Modificare la funzione conta dell'esempio 22 in modo che essa fornisca come risultato il numero di atomi distinti che occorrono in una formula:*

$$conta : \mathcal{F} \mapsto \mathbb{N}$$

Esercizio 47 *Definire una funzione ricorsiva su $\mathcal{F}$*

$$sottof : \mathcal{F} \mapsto 2^{\mathcal{F}}$$

che assegna a ogni formula A l'insieme $sottof(A)$ delle sottoformule di A. Provare che se A ha n connettivi allora $sottof(A)$ contiene al più $2n + 1$ sottoformule.

Esercizio 48 *Utilizzare il teorema 2.4 per realizzare un algoritmo che, ricevendo in ingresso una formula in cui compaiono solo i connettivi $\wedge, \rightarrow, \neg$, lettere dell'alfabeto italiano e parentesi, determini se la formula è ben formata o no. L'algoritmo deve costruire l'albero sintattico associato alla formula, ovvero l'albero che ha la radice etichettata con il* connettivo principale *della formula da analizzare e ciascun nodo, tranne le foglie, è etichettato da un connettivo. Ad esempio, il connettivo principale di $((A \wedge B) \rightarrow (C \rightarrow A))$ è la prima occorrenza del connettivo $\rightarrow$.*

2.2 Sistemi di valutazione

Abbiamo visto nel paragrafo 2.1 che il linguaggio di una logica è costruito a partire da elementi iniziali (che abbiamo chiamato $\mathcal{A}tom$ e abbiamo lasciato non specificati), per mezzo di operatori unari e binari. Il modo in cui abbiamo costruito l'insieme delle formule ben formate ci garantisce una proprietà chiamata *composizionalità* del linguaggio che, come vedremo fra breve, è molto importante nel momento in cui si voglia associare un significato alle formule[2].

Normalmente, ogni linguaggio (pensiamo all'italiano piuttosto che all'inglese o a un linguaggio di programmazione come il Pascal) associa al suo insieme di simboli iniziali un significato. Ad esempio in Pascal l'insieme dei simboli iniziali è l'insieme delle parole riservate $\{$**while**, **until**, $\ldots\}$ e a ciascun simbolo è associato un comando. Così in italiano, come in tutte le lingue parlate, possiamo considerare le parole (il vocabolario), come l'insieme dei simboli iniziali e, ovviamente, ogni parola assume un significato rispetto al mondo reale. Tuttavia, quando mettiamo insieme le parole e con esse costruiamo una frase, per mezzo di opportuni costruttori linguistici, l'interpretazione della frase dipende non solo dal significato delle parole ma dalla costruzione stessa della frase.

Esempio 24 *Considerare le due frasi "Tutti i bambini sono buoni" e "Tutti i buoni sono bambini".*

Pertanto, quando parliamo di "interpretazione" di un linguaggio ci riferiamo sia al significato associato ai suoi simboli iniziali sia all'estensione di tale significato, per mezzo dei costruttori del linguaggio, alle frasi del linguaggio. In tal modo anche l'interpretazione ha un carattere composizionale.

Se il linguaggio appartiene a un sistema formale, individuare il significato dei simboli iniziali ed estendere tale significato, mediante i costruttori del linguaggio, alle formule ben formate, vuol dire fornire il sistema formale di una *semantica*. In qualche modo la semantica offre un apparato interpretativo che chiamiamo *sistema di valutazione* della logica.

Definizione 2.6 *Un sistema di valutazione $\mathcal{S}$ è una tripla $\langle \mathcal{B}, \mathcal{T}, \mathcal{O}p \rangle$, dove:*

1. *$\mathcal{B}$ è un insieme detto* insieme dei valori di verità *che contiene almeno due elementi.*
2. *$\mathcal{T} \subset \mathcal{B}$ è un sottoinsieme proprio di $\mathcal{B}$, detto* insieme designato a rappresentare il vero.
3. *$\mathcal{O}p = \{\mathcal{O}p_\star, \mathcal{O}p_\circ\}$ è l'insieme delle funzioni che corrispondono ai costruttori del linguaggio $\{\star, \circ\}$, con $\mathcal{O}p_\star : \mathcal{B} \mapsto \mathcal{B}$ e $\mathcal{O}p_\circ : \mathcal{B} \times \mathcal{B} \mapsto \mathcal{B}$. $\mathcal{O}p_\circ$ e $\mathcal{O}p_\star$ interpretano $\circ$ e $\star$.*

[2] Osserviamo che la proprietà di composizionalità vale per ogni insieme induttivo.

Solitamente $\mathcal{B}$ contiene esattamente due elementi $\mathcal{B}= \{\mathbf{t}, \mathbf{f}\}$ (e in tal caso la logica è detta "a due valori di verità": $\mathbf{t}$ sta per il vero, $\mathbf{f}$ sta per il falso), ma può anche contenerne tre, quattro, o un qualunque numero finito o infinito (anche non numerabile, come nel caso delle logiche "fuzzy");

$\mathcal{T}$ contiene l'insieme dei valori che designano il vero. Nel caso della logica a due valori di verità, cioè nel caso $\mathcal{B}= \{\mathbf{t}, \mathbf{f}\}$, si avrà $\mathcal{T}= \{\mathbf{t}\}$.

Per ciascun operatore in $\mathcal{O}p$ viene definita una *tabella di verità* che dipende dai valori di verità definiti nel sistema di valutazione.

Esempio 25 *Supponiamo che* $\circ$ *sia l'implicazione materiale* $\rightarrow$*, e siano* $\mathcal{S}_2$ *e* $\mathcal{S}_3$ *due sistemi di valutazione. In* $\mathcal{S}_2$*,* $\mathcal{B}_2 = \{\mathbf{t}, \mathbf{f}\}$ *e in* S_3*,* $\mathcal{B}_3 = \{\mathbf{t}, \mathbf{u}, \mathbf{f}\}$*, dove* $\mathbf{u}$ *vuol dire indefinito. Definiamo le tabelle di verità per* $\mathcal{O}p_{\rightarrow}$ *come segue:*

$\mathcal{O}p_{\rightarrow}$	$\mathbf{t}$	$\mathbf{f}$
$\mathbf{t}$	$\mathbf{t}$	$\mathbf{f}$
$\mathbf{f}$	$\mathbf{t}$	$\mathbf{t}$

1. Tabella per $\mathcal{O}p_{\rightarrow}$ a due valori

$\mathcal{O}p_{\rightarrow}$	$\mathbf{t}$	$\mathbf{f}$	$\mathbf{u}$
$\mathbf{t}$	$\mathbf{t}$	$\mathbf{f}$	$\mathbf{u}$
$\mathbf{f}$	$\mathbf{t}$	$\mathbf{t}$	$\mathbf{t}$
$\mathbf{u}$	$\mathbf{t}$	$\mathbf{f}$	$\mathbf{u}$

2. Tabella per $\mathcal{O}p_{\rightarrow}$ a tre valori

Osserviamo che gli indici di riga e colonna sono, rispettivamente, i valori del primo e secondo argomento di $\mathcal{O}p_{\rightarrow}$ e gli elementi della matrice sono i corrispondenti valori.

Il sistema di valutazione non dice come valutare gli elementi di $\mathcal{A}tom$, per questo definiamo un'*assegnazione*:

Definizione 2.7 *Dato un sistema di valutazione* $\mathcal{S} = \langle \mathcal{B}, \mathcal{T}, \mathcal{O}p \rangle$ *per* $\mathcal{L}$*, un'assegnazione* $\mathcal{V}$ *è una funzione:*

$$\mathcal{V} : \mathcal{A}tom \rightarrow \mathcal{B}$$

Estendiamo l'assegnazione $\mathcal{V}$ a una interpretazione $I_{\mathcal{V}} : \mathcal{F} \mapsto \mathcal{B}$ per le formule di $\mathcal{F}$ come segue.

Definizione 2.8 *Sia* $\mathcal{V}$ *un'assegnazione, una* interpretazione $I_{\mathcal{V}} : \mathcal{F} \mapsto \mathcal{B}$ *è definita come:*

- $I_{\mathcal{V}}(A) = \mathcal{V}(A)$ *se* $A \in \mathcal{A}tom$*;*
- $I_{\mathcal{V}}(\star A) = \mathcal{O}p_{\star}(I_{\mathcal{V}}(A))$
- $I_{\mathcal{V}}(A \circ B) = \mathcal{O}p_{\circ}(I_{\mathcal{V}}(A), I_{\mathcal{V}}(B))$

Una interpretazione è quindi composta da un'assegnazione alle formule atomiche e da un sistema di valutazione che caratterizza il modo in cui vengono valutati gli operatori nella logica. Pertanto, fissato un sistema di valutazione, l'interpretazione di una formula di fatto dipende soltanto dalla assegnazione dei valori di verità alle formule atomiche.

Esempio 26 *Supponiamo che l'operatore* $\circ$ *sia la congiunzione* $\wedge$ *e la tabella di verità di* $\wedge$*, per il sistema* $\mathcal{S}_2 = \langle \mathcal{B}, \mathcal{T}, \mathcal{O}p \rangle$ *con* $\mathcal{B} = \{\mathbf{t}, \mathbf{f}\}$ *e* $\mathcal{T} = \{\mathbf{t}\}$*, sia la seguente:*

$\mathcal{O}p_\wedge$	t	f
t	t	f
f	f	f

Sia $A = (B \wedge (C \wedge D))$ *e sia* $I_\mathcal{V}(B) = \mathcal{V}(B) = \mathbf{t}$*,* $I_\mathcal{V}(C) = \mathcal{V}(C) = \mathbf{f}$ *e* $I_\mathcal{V}(D) = \mathcal{V}(D) = \mathbf{t}$*. Si può verificare che* $I_\mathcal{V}(A) = \mathbf{f}$*.*

È chiaro che $I_\mathcal{V}$ è definita ricorsivamente e, quindi, poiché abbiamo mostrato (teorema 2.4) che l'insieme $\mathcal{F}$ è liberamente generato, per il teorema di ricorsione segue che $I_\mathcal{V}$ estende $\mathcal{V}$ ed è unica. Tuttavia per assicurarcene enunciamo il seguente:

Lemma 2.9 *Per ogni formula* A *di* $\mathcal{F}$*, per ogni assegnazione* $\mathcal{V}$ *e* $\mathcal{V}'$ *tale che* $\mathcal{V}(P) = \mathcal{V}'(P)$*, per ogni* $P \in \mathcal{A}tom$ *che occorre in* A*,* $I_\mathcal{V}(A) = I_{\mathcal{V}'}(A)$*.*

Dimostrazione Lasciata al lettore (si veda l'esercizio 49), si suggerisce di usare l'induzione strutturale. □

Definizione 2.10 *Una formula* A *è una* tautologia *rispetto a un sistema di valutazione* $\mathcal{S}$ *sse* $I_\mathcal{V}(A) \in \mathcal{T}$ *per ogni interpretazione* $I_\mathcal{V}$*.*

Dire che $I_\mathcal{V}(A) \in \mathcal{T}$ per ogni assegnazione $\mathcal{V}$ significa dire che lo è per ogni interpretazione $I_\mathcal{V}$.

Definizione 2.11 *Una formula* A *è* soddisfacibile *se esiste una interpretazione* $I_\mathcal{V}$*, tale che* $I_\mathcal{V}(A) \in \mathcal{T}$*.*

Come precedentemente, cerchiamo un'assegnazione di verità per le formule atomiche di A, tale che $I_\mathcal{V}(A) \in \mathcal{T}$.

Definizione 2.12 *Una formula* A *è* insoddisfacibile *o* contraddittoria *se* $I_\mathcal{V}(A) \notin \mathcal{T}$ *per ogni* $I_\mathcal{V}$*.*

Quindi, se una formula è una tautologia, allora essa è anche soddisfacibile; se una formula non è una tautologia essa può essere sia soddisfacibile che non. Se la logica ha un operatore unario $\neg$ per denotare la negazione, avremo che la negazione $\neg A$ di una tautologia A è una formula insoddisfacibile, cioè una contraddizione.

Alternativamente alle funzioni di interpretazione, si può studiare il significato di una formula in una "struttura" matematica o "realtà" $\models$: invece di una notazione funzionale (ovvero $I_\mathcal{V}$) si adotta una notazione relazionale, basata sulla relazione di *soddisfacibilità* $\models$, dove $\models \subseteq \mathcal{M} \times \mathcal{F}$; $\mathcal{M}$ è una struttura e $\mathcal{M} \models A$ si legge "$\mathcal{M}$ modella A".

Rimandiamo una definizione più dettagliata a quando presenteremo le varie logiche, ovvero daremo un carattere più preciso ai costruttori del linguaggio e alle strutture.

2.2.1 Esercizi

Esercizio 49 *Dimostrare il lemma 2.9.*

Esercizio 50 *Siano $\mathcal{S}_2$ e $\mathcal{S}_3$ come nell'esempio 25 e $\circ$ sia l'implicazione materiale con associate le due tabelle di verità definite nell'esempio 25.*

Sia $A = (B \to (C \to B))$. Verificare che in S_2, per qualunque assegnazione agli atomi, l'interpretazione $I_{\mathcal{V}}(A) = \mathbf{t}$. Cosa succede con S_3?

Esercizio 51 *Definire un insieme di connettivi O e delle funzioni $\mathcal{O}p$ che li interpretano, con relative tabelle di verità associate. Definire ricorsivamente la relazione $\models$ per i connettivi. Data una formula A, sia $\mathcal{V}(P) = \mathbf{t}$ sse $P \in M$, per ogni atomo P che occorre in A. Verificare che $I_{\mathcal{V}}(A) = \mathbf{t}$ sse $M \models A$.*

Esercizio 52 *Definire i concetti di* indefinito *e* sovradefinito. *Associare a essi dei valori di verità da unire ai due valori* $\mathbf{t}$ *e* $\mathbf{f}$. *Costruire delle tabelle per opportuni connettivi, nello stile delle tabelle fornite nell'esempio 25.*

2.3 Apparato deduttivo

Vediamo ora come è possibile manipolare un linguaggio Λ utilizzando categorie formali o nozioni meta-logiche, come regole di inferenza e assiomi, e mediante questi definire un sottoinsieme interessante di formule ben formate, cioè quello dei teoremi di Λ. Introdurremo prima la nozione di dimostrazione, poi quella di teorema.

Le regole di inferenza di $\mathcal{R}$ spesso vengono scritte nella seguente forma:

$$\frac{A_1 \cdots A_n}{A}$$

dove le formule scritte al di sopra della riga orizzontale vengono chiamate *premesse* della regola e la formula scritta al di sotto della riga è detta la sua *conclusione*.

L'insieme Ax degli assiomi di una logica Λ può essere vuoto, finito o infinito. Particolarmente interessante è il caso in cui Ax è finito o descrivibile in maniera finita, cioè è decidibile.

Talvolta gli assiomi vengono scritti come regole di inferenza senza premesse, cioè se $A \in Ax$ scriveremo:

$$\frac{}{A}$$

Definizione 2.13 *Una sequenza finita di formule $A_1, \cdots, A_n$ di Λ è detta una* dimostrazione *o* prova *in Λ se, per ogni i compreso tra 1 ed n, o $A_i \in Ax$, cioè è un assioma di Λ, oppure è una conseguenza diretta mediante una delle regole di $\mathcal{R}$ di alcune delle formule che la precedono nella sequenza.*

Definizione 2.14 *Una formula A di Λ è detta un* teorema *di Λ se esiste una dimostrazione di Λ che ha A come ultima formula. Tale dimostrazione è detta* una dimostrazione di A *in* Λ.

Indicheremo con $\vdash A$ il fatto che A è un teorema di Λ. In caso di ambiguità scriveremo $\vdash_\Lambda A$ oppure $\vdash_{\mathcal{R}} A$ a sottolineare che A ha una dimostrazione in Λ, oppure che A ha una dimostrazione mediante regole di $\mathcal{R}$. $\vdash_\Lambda A$ si legge anche: A è derivabile in Λ; $\vdash_{\mathcal{R}} A$ si legge anche: A è derivabile mediante $\mathcal{R}$.

Definizione 2.15 *Sia Γ un insieme di formule, cioè $\Gamma \subseteq \mathcal{F}$. Diciamo che una formula A è una conseguenza di Γ (lo scriviamo $\Gamma \vdash A$, che si legge "A si deriva da Γ") se esiste una sequenza di formule $A_1, \ldots, A_n$ tale che A è A_n e per ciascun i compreso tra 1 e n si ha che o $A_i \in Ax$, o $A_i \in \Gamma$ o A_i è conseguenza diretta di alcune delle formule che la precedono nella sequenza.*

Tale sequenza è detta una *derivazione* o *prova di A da Γ in Λ*. Gli elementi di Γ sono detti le *premesse*, o *ipotesi*, o anche *assunzioni* di A. Se Γ è un insieme finito $B_1, \ldots, B_n$ scriviamo indifferentemente $\{B_1, \ldots, B_n\} \vdash A$ oppure $B_1, \ldots, B_n \vdash A$.

Si noti come un teorema altro non è se non una formula derivabile "per via di logica", cioè una derivazione da un insieme di assunzioni Γ vuoto: scrivere $\vdash A$ equivale a scrivere $\emptyset \vdash A$.

Definizione 2.16 *Sia Γ un insieme di formule. La* chiusura deduttiva *di Γ, denotata con $Cn(\Gamma)$, è l'insieme di tutte le formule che sono conseguenza di Γ (cioè $Cn(\Gamma) = \{A | \Gamma \vdash A\}$)*

Definizione 2.17 *Sia Γ un insieme di formule.* Γ *è* consistente *sse $Cn(\Gamma)$ è un sottoinsieme proprio di $\mathcal{F}$ (cioè $Cn(\Gamma) = \{A | \Gamma \vdash A\} \subset \mathcal{F}$).*

La nozione di conseguenza in una logica Λ può godere di una o più delle seguenti proprietà:

- Se $A \in \Gamma$ allora $\Gamma \vdash A$, o anche se $A \in \Gamma$ allora $A \in Cn(\Gamma)$, dunque $\Gamma \subseteq Cn(\Gamma)$ (questa proprietà è detta *inclusione*);
- Se $\Gamma \subseteq \Delta$ e $\Gamma \vdash A$ allora $\Delta \vdash A$, o anche $Cn(\Gamma) \subseteq Cn(\Delta)$ (questa proprietà è detta *monotònia*);
- $\Gamma \vdash A$ se e solo se esiste un sottoinsieme finito Δ di Γ tale che $\Delta \vdash A$ (questa proprietà è detta *compattezza*);
- Se $\Delta \vdash A$ e per ogni B di Δ si ha che $\Gamma \vdash B$ allora si ha che $\Gamma \vdash A$ (questa proprietà è detta *taglio sulle premesse*).

In una logica in cui sia presente un operatore di implicazione, può inoltre valere un *teorema di deduzione*, cioè:

$$\Gamma \vdash A \to B \text{ sse } \Gamma, A \vdash B.$$

Notare che in logica un *teorema* è una formula derivabile *nel* linguaggio. Una *proprietà del* linguaggio e delle sue formule non è quindi un teorema, ma un *meta-teorema*. Le proprietà che abbiamo sopra enunciato, nel caso valgano per una logica Λ, sono meta-teoremi per Λ.

Nel seguito la parola "dimostrazione" verrà usata con un duplice significato: una "dimostrazione" in una logica Λ è una sequenza di formule di Λ che porta ad un teorema di Λ (coerentemente con le definizioni 2.13 e 2.14); una "dimostrazione" di un meta-teorema che concerne una logica Λ è una sequenza di argomentazioni usata per provare una proprietà di cui la logica Λ gode.

2.3.1 Esercizi

Esercizio 53 *Nella definizione di logica data in 2.1, si richiede che ciascuna regola di inferenza sia decidibile. Analizzare questa scelta: quali sarebbero le conseguenze nel caso non si ponesse questa condizione?*

Esercizio 54 *Si supponga di avere un insieme* $\Delta = \{A, A \to B, B \to C\}$ *in una logica in cui valgano tutte le proprietà elencate sopra (inclusione, monotònia, eccetera) e in cui nell'apparato deduttivo ci sia solo una regola* $MP(A, A \to B, B)$. *Dire cosa può essere sostituito a* ? *in* $\Delta \vdash ?$. *Si individui un insieme* Γ *su cui si possa applicare il* taglio sulle premesse.

2.4 Considerazioni sui sistemi formali

In genere le formule di un insieme Γ sono intese rappresentare una data realtà. In questo caso tutte le strutture in cui le formule di Γ sono verificate sono dette *modelli di* Γ e l'insieme Γ è detto l'insieme degli *assiomi propri* associati a tali modelli. Quindi, rappresentare una data realtà in un sistema formale Λ significa individuare un insieme di formule di Λ (gli assiomi propri) che hanno quella realtà a modello, non sempre – anzi di rado – modello esclusivo.

Abbiamo visto che una logica si può definire tramite un linguaggio, un insieme di assiomi e un insieme di regole di inferenza. Stabilire, per mezzo degli assiomi e delle regole di inferenza, quali sono le formule "interessanti" della logica, ovvero i teoremi, significa fare "deduzione". I passi necessari a stabilire se una formula è un teorema della logica possono essere espressi tramite un algoritmo e quindi meccanizzati. Possiamo dire che, in questo modo, stiamo utilizzando il sistema formale come un *sistema di deduzione* e, quindi, non siamo effettivamente interessati al significato delle formule del linguaggio ma, piuttosto, al modo in cui mettiamo insieme tali formule per costruire una prova diretta oppure, come vedremo spesso in

seguito, una *refutazione*, cioè una prova che la negazione di una certa formula è in contraddizione con gli assiomi e le ipotesi fatte.

Il fatto che i passi di una dimostrazione possano essere posti in forma di un algoritmo e meccanizzati non significa che possiamo sempre avere una risposta se una formula è un teorema oppure no, né significa che per costruire una prova non occorra alcuna creatività. Che una procedura di prova *termini*, con una risposta positiva o negativa (a seconda che la formula in questione sia un teorema o meno) dipende dalla logica. Se tale procedura esiste, cioè se l'insieme dei teoremi di una logica è decidibile, diremo che *la logica è decidibile*.

Abbiamo visto che un linguaggio ha un significato. Possiamo essere interessati a un sistema formale dal punto di vista "del significato", invece che da un punto di vista deduttivo. In altri termini il linguaggio può servirci per caratterizzare delle strutture. In tal caso siamo interessati alla *teoria dei modelli*. L'aspetto interessante della teoria dei modelli è che ci porta a dare una caratterizzazione precisa di cos'è la nozione di *verità* in una struttura. Abbiamo visto che la nozione di verità alla quale siamo interessati, deve ricalcare gli stessi principi di composizionalità del linguaggio, ovvero la verità (o falsità) di una formula si ottiene tramite la verità (o falsità) delle sue componenti (sottoformule), e questo ci è assicurato dalle tabelle di verità per gli operatori della logica.

È chiaro che così come si possono definire algoritmi associati alle procedure di prova, si possono definire algoritmi associati alla interpretazione di una formula; algoritmi che possono stabilire se una formula è vera in tutte le strutture della logica o meno. Analogamente alle procedure di prova, la possibilità di stabilire effettivamente la validità di una formula, dipende dal linguaggio.

È naturale chiedersi che tipo di connessione c'è fra i due modi di utilizzare un sistema formale: quello legato alle dimostrazioni e quello semantico legato ai modelli. La connessione è auspicabile nel senso che sarebbe per lo meno strano riuscire a provare teoremi e non riuscire a collegarli alle formule valide. Tutto ciò però non è ovvio, nel senso che procedure di deduzione e procedure associate alla validità non sono necessariamente complementari l'una all'altra.

Ad esempio potremmo avere un insieme infinito di strutture da analizzare per verificare se una certa formula è vera in esse, ma solo un insieme finito di assiomi che descrive tali strutture. E da tale insieme potremmo avere una procedura di prova che ci fornisce una risposta.

Nel primo caso, l'arbitrarietà con cui si può scegliere un modello per un insieme di formule lascia poco spazio alla decidibilità della nozione di validità.

In altre condizioni, potremmo essere interessati a trovare solo un modello per una formula, cioè a verificarne la soddisfacibilità, mentre la procedura di prova non è in grado di darci una risposta se quella formula è un teorema o meno. Vedremo in seguito che esistono classi di formule per le quali stabilire quali sono valide si riduce alla scelta di un unico modello.

In effetti è proprio l'interazione tra teoria dei modelli e teoria della deduzione che rende un sistema formale interessante: ciò che si può manipolare sintatticamente ha

un effettivo riscontro, in termini di nozione di verità, nella realtà in cui il linguaggio è interpretato.

Tale interazione è stabilita dalle seguenti definizioni:

Definizione 2.18 *Un apparato deduttivo* $\mathcal{R}$ *è* corretto *se per ogni formula* $A \in \mathcal{F}$, $\vdash_{\mathcal{R}} A$ *implica* $\models A$.

Definizione 2.19 *Un apparato deduttivo* $\mathcal{R}$ *è* completo *rispetto a una classe di formule* $\Gamma \subseteq \mathcal{F}$, *se per ogni formula* $A \in \Gamma$, $\models A$ *implica* $\vdash_{\mathcal{R}} A$.

Risultano di particolare interesse le logiche Λ e le definizioni di semantiche per cui si possa dimostrare un teorema di correttezza e completezza, ossia una coincidenza tra la nozione semantica di conseguenza (cioè quella basata sui modelli) e quella sintattica (basata sulle dimostrazioni). Tale meta-teorema talvolta si esemplifica scrivendo: $\vdash \equiv \models$.

Il teorema di correttezza e completezza stabilisce una stretta relazione tra il livello sintattico (cioè della deduzione) e quello semantico (cioè dei modelli) per un sistema formale, e stabilisce una stretta analogia tra le nozioni introdotte ai due livelli. Tale relazione si può riassumere nella seguente tabella, cui è opportuno fare riferimento nei prossimi capitoli:

sintassi	semantica
$\Gamma \vdash_{\mathcal{R}} A$ derivabilità	$\Gamma \models A$ conseguenza logica
$\vdash_{\mathcal{R}} A$ teorema	$\models A$ validità
$Cn(\Gamma) \subset \mathcal{F}$ consistenza	$\Gamma \not\models \mathbf{f}$ soddisfacibilità
inconsistenza $Cn(\Gamma) = \mathcal{F}$	insoddisfacibilità $\Gamma \models \mathbf{f}$

Chiudiamo questo capitolo con una riflessione sull'uso della parola "modello" in logica. Nel linguaggio comune, così pure come in quello scientifico, la parola modello ha due distinti e opposti significati; entrambi hanno a che vedere con una rappresentazione e ciò che è rappresentato. Il problema è che la parola modello è talvolta usata per indicare il primo termine della relazione (cioè la rappresentazione) talvolta il secondo (cioè l'oggetto rappresentato). Così per esempio si dice che il manufatto a scala ridotta che rappresenta una barca o un aereo è un modello della barca o dell'aereo. Anche in economia si parla di modello per intendere una rappresentazione matematica del mercato. In logica matematica al contrario, la rappresentazione è chiamata teoria e ciò che è rappresentato è chiamato modello. Questo uso è contrario a quello scientifico, ma coincide con quello degli artisti, per

esempio i pittori, che parlano di modello per intendere ciò che viene rappresentato, cioè l'oggetto della pittura (che ne è quindi la rappresentazione).

2.4.1 Esercizi

Esercizio 55 *Stabilire le relazioni di contenimento e complementarietà tra gli insiemi delle formule valide, soddisfacibili e insoddisfacibili.*

Esercizio 56 *Individuare alcuni costruttori linguistici della lingua italiana.*

2.5 Riepilogo

In questo capitolo abbiamo introdotto le definizioni relative ai sistemi formali, o sistemi logici. In particolare abbiamo definito:

1. il linguaggio, cioè l'insieme delle formule ben formate;
2. la definizione induttiva di formula e il relativo principio di induzione e ricorsione strutturale;
3. la nozione di sistema di valutazione: il modo di attribuire un significato alle formule della logica;
4. la nozione di insieme di valori di verità e di insieme designato a rappresentare il vero;
5. la nozione di assegnazione e di valutazione di una formula;
6. la nozione di tautologia, formula soddisfacibile insoddisfacibile e contraddittoria;
7. la nozione di modello per un insieme di formule e di formula valida;
8. l'apparato deduttivo, cioè l'insieme delle regole di inferenza;
9. la nozione di dimostrazione o prova, e di teorema in una logica;
10. la nozione di chiusura deduttiva e di consistenza;
11. le proprietà di inclusione, monòtonia, compattezza e taglio sulle premesse;
12. il teorema di deduzione.

Abbiamo infine correlato la nozione sintattica di deduzione $\vdash$ con quella semantica di conseguenza logica $\models$, introducendo le proprietà di correttezza e completezza di un apparato deduttivo.

Parte II

La logica classica

In questa parte presentiamo la logica classica: proposizionale e del primo ordine.

Nel capitolo 3 introduciamo il più semplice linguaggio logico: La *logica proposizionale*, o *calcolo delle proposizioni*, ne presentiamo la sintassi, la semantica e ne discutiamo la decidibilità.

Nel capitolo 4 ci occupiamo di *deduzione* per la logica proposizionale: presentiamo i sistemi assiomatici, la deduzione naturale, il metodo dei tableau e la risoluzione. Presentiamo quindi, nel capitolo 5, la nozione di teoria proposizionale.

Nel capitolo 6 presentiamo il linguaggio e le strutture di interpretazione per la *logica del primo ordine*, o *calcolo dei predicati*. Quindi, nei capitoli 7 e 8 introduciamo la deduzione nel caso predicativo, ripercorrendo quanto introdotto nel capitolo 4, questa volta nel caso più interessante della logica del primo ordine. Ci soffermiamo, infine, nel capitolo 9 sulle teorie del primo ordine.

Capitolo 3
La logica proposizionale

Affrontiamo in questo capitolo la logica proposizionale. Il potere espressivo di questo formalismo è limitato: in esso infatti possiamo esprimere proposizioni che sono vere o false, ma non proprietà che possono valere o no su uno/molti/tutti gli individui di una certa classe. Col basso potere espressivo si coniuga la decidibilità di questa logica. Presenteremo la sintassi e la semantica del calcolo proposizionale, mentre il problema della deduzione verrà affrontato nei capitoli successivi.

3.1 Sintassi

Introduciamo in questo paragrafo la nozione di *linguaggio proposizionale* $\mathcal{L}_{\mathcal{W}}$ costruito su un alfabeto $\mathcal{W}$. Iniziamo con la definizione di alfabeto.

Definizione 3.1 *Un* alfabeto $\mathcal{W}$ *è costituito da:*

- *I connettivi proposizionali* $\neg$ *(unario) e* $\wedge$, $\vee$, $\rightarrow$ *e* $\leftrightarrow$ *(binari);*
- *Le costanti proposizionali* $\top$, $\bot$ *(per denotare il vero e il falso);*
- *Un insieme non vuoto (finito o numerabile) di simboli proposizionali* $\mathcal{P} = \{A, B, \ldots, P, Q, \ldots\}$*;*
- *I simboli separatori '(' e ')'.*

Nel seguito scriveremo $\mathcal{L}$ quando $\mathcal{W}$ è chiaro dal contesto. Definiamo ora le formule di $\mathcal{L}$.

Definizione 3.2 *L'insieme* PROP *delle* formule ben formate *o* formule *del linguaggio proposizionale* $\mathcal{L}$ *è l'insieme definito induttivamente come segue:*

1. *Le costanti e i simboli proposizionali sono formule;*
2. *Se A è una formula $(\neg A)$ è una formula;*
3. *Se $\circ$ è un connettivo binario (cioè $\circ \in \{\vee, \wedge, \rightarrow, \leftrightarrow\}$)e se A e B sono due formule, $(A \circ B)$ è una formula.*

Le costanti e i simboli proposizionali sono anche detti *atomi*, le loro negazioni sono dette *atomi negati*. Gli atomi e gli atomi negati sono anche detti *letterali*. Gli atomi negati sono talvolta detti *letterali negativi*. Una formula del linguaggio proposizionale è anche detta *proposizione* o *enunciato proposizionale*.

Data una formula A di PROP, ogni formula B che appare come componente di A è detta *sottoformula* di A. La definizione di *sottoformula*, di *connettivo principale* e di *sottoformula immediata* è la seguente:

Definizione 3.3 *Sia A una formula di* PROP, *l'insieme delle* sottoformule *di A è definito come segue:*

1. *Se A è una costante o un simbolo proposizionale allora A stessa è la sua sola sottoformula.*
2. *Se A è una formula del tipo $(\neg A')$ allora le sottoformule di A sono A stessa e le sottoformule di A'; $\neg$ è detto* connettivo principale *e A'* sottoformula immediata *di A.*
3. *Se A è una formula del tipo $B \circ C$ dove $\circ$ è un connettivo binario (cioè $\circ \in \{\vee, \wedge, \rightarrow, \leftrightarrow\}$), e B e C due formule, le sottoformule di A sono A stessa e le sottoformule di B e C; $\circ$ è detto* connettivo principale*; B e C* sottoformule immediate *di A.*

La definizione che abbiamo introdotto per le formule proposizionali fa un abbondante uso di parentesi. Come abbiamo detto nel capitolo precedente, le parentesi si possono eliminare con l'introduzione di una opportuna precedenza tra i connettivi. Per le formule proposizionali si usa la seguente convenzione: la massima precedenza a $\neg$, poi, nell'ordine, ai connettivi $\wedge$,$\vee$, $\rightarrow$ e infine $\leftrightarrow$. Questo significa che, in assenza di parentesi, una formula ben formata va parentetizzata privilegiando le sottoformule i cui connettivi principali hanno precedenza più alta. A parità di precedenza, cioè se siamo in presenza di più occorrenze dello stesso connettivo, si assume che esso associ a destra.

Esempio 27

La formula $\neg A \wedge \neg B$
viene parentetizzata come $((\neg A) \wedge (\neg B))$.

La formula $A \wedge B \vee C$
viene parentetizzata come $((A \wedge B) \vee C)$.

La formula $A \rightarrow B \rightarrow C$
viene parentetizzata come $(A \rightarrow (B \rightarrow C))$.

La formula $\neg A \wedge \neg B \rightarrow C \wedge D \wedge E$
viene parentetizzata come $(((\neg A) \wedge (\neg B)) \rightarrow (C \wedge (D \wedge E)))$.

La formula $\neg A \wedge (\neg B \rightarrow C) \wedge D \wedge E$
viene parentetizzata come $((\neg A) \wedge ((\neg B) \rightarrow C) \wedge (D \wedge E))$.

3.1.1 Esercizi

Esercizio 57 *Verificare che i seguenti risultati mostrati nel capitolo precedente valgono per la logica proposizionale.*

1. *Ogni proposizione in* PROP *ha lo stesso numero di parentesi '(' e ')';*
2. PROP *è un insieme liberamente generato.*

Esercizio 58 *Definire ricorsivamente su* PROP *la funzione*

$$simb: \text{PROP} \mapsto 2^{\mathbf{P}}$$

che restituisce l'insieme dei simboli proposizionali che occorrono in una proposizione.

Esercizio 59 *Definire una procedura ricorsiva di visita dell'albero sintattico associato a una espressione proposizionale. La procedura deve:*

a) stabilire se l'espressione è una formula ben formata o meno;

b) nel caso lo sia, stampare la formula proposizionale opportunamente parentetizzata.

3.2 Semantica

Il *sistema di valutazione* $\mathcal{S} = \langle \mathcal{B}, \mathcal{T}, \mathcal{O}p \rangle$ della logica proposizionale è definito da:

1. $\mathcal{B}= \{0, 1\}$;
2. $\mathcal{T} = \{1\}$;
3. $\mathcal{O}p= \{\mathcal{O}p_{\neg}, \mathcal{O}p_{\wedge}, \mathcal{O}p_{\vee}, \mathcal{O}p_{\rightarrow}, \mathcal{O}p_{\leftrightarrow}\}$ uno per ogni connettivo del linguaggio $\{\neg, \wedge, \vee, \rightarrow, \leftrightarrow\}$, con $\mathcal{O}p_{\neg} : \mathcal{B} \mapsto \mathcal{B}$ e $\mathcal{O}p_{\circ} : \mathcal{B} \times \mathcal{B} \mapsto \mathcal{B}$, $\circ \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$.

La funzione $\mathcal{O}p_{\neg}$ della logica proposizionale è definita come segue: $\mathcal{O}p_{\neg}(1) = 0$ e $\mathcal{O}p_{\neg}(0) = 1$. Questo è di solito espresso in maniera concisa come: $\neg(0) = 1$ e $\neg(1) = 0$, cioè la funzione di valutazione associata a un connettivo viene indicata col simbolo stesso del connettivo e viene definita in forma tabellare mediante la *tavola* o *tabella dei valori di verità* per il connettivo:

	$\neg$
1	0
0	1

Evitando di presentare le funzioni $\mathcal{O}p$ della logica proposizionale per i connettivi binari, le presentiamo subito in forma di tabelle dei valori di verità.

		$\wedge$	$\vee$	$\rightarrow$	$\leftrightarrow$
1	1	1	1	1	1
1	0	0	1	0	0
0	1	0	1	1	0
0	0	0	0	1	1

Il connettivo di *congiunzione* $\wedge$ viene definito in modo che $A \wedge B$ è vero sse sia A che B (i due *congiunti*) sono veri, quindi $\mathcal{O}p_{\wedge} = min$. Il connettivo di *disgiunzione* $\vee$ viene definito in modo che $A \vee B$ è vero sse A oppure B (uno dei due *disgiunti*) sono veri, quindi $\mathcal{O}p_{\vee} = max$. Si noti che con questa definizione $\vee$ è la disgiunzione inclusiva, cioè corrisponde al "vel" latino e non all'"aut".

La definizione della semantica del connettivo di *implicazione* $A \rightarrow B$ (detta *implicazione materiale*, in cui A è detto *premessa* e B *conseguenza*) è in un certo senso meno intuitiva. Innanzi tutto si noti che, con la definizione data, si ha che $A \rightarrow A$ è sempre vero, qualunque sia il valore di verità di A; questo corrisponde alla nostra intuizione. Possiamo quindi accettare il fatto che affinché $A \rightarrow B$ sia vero basta che B sia vero, indipendentemente dal valore di verità di A. Questo di fatto ci dice che, se B è vero e $B \rightarrow B$ è vero, possiamo "rafforzare" la premessa sostituendo a B un qualunque A e l'implicazione resta vera. Ovviamente, se la premessa è vera e la conseguenza è falsa, l'implicazione è falsa. Si noti che con questa definizione è difficile immaginare un nesso di causa-effetto tra premessa e conseguenza. Infatti in un opportuno linguaggio proposizionale avremmo anche che "Se il Presidente della Repubblica si chiama Filippo, allora oggi ho vinto alla Lotteria di Capodanno" è una implicazione vera, anche se sia la premessa che la conseguenza sono false (a oggi, marzo 2000) ed è difficile immaginare un nesso di causa-effetto tra le due.

La definizione della semantica del connettivo di *doppia implicazione* è del tutto intuitiva: il valore di verità di $A \leftrightarrow B$ è vero se i valori di verità di A e B coincidono.

Definizione 3.4 *Un'*assegnazione booleana $\mathcal{V}$ *ai simboli proposizionali* $\mathcal{P}$ *è una funzione totale:*

$$\mathcal{V} : \mathcal{P} \rightarrow \{1, 0\}.$$

Analogamente a quanto osservato nel capitolo 2, per il teorema di ricorsione e per il fatto che l'insieme delle formule proposizionali è liberamente generato, ogni asse-

gnazione $\mathcal{V}$ si estende a un'unica interpretazione o valutazione booleana.

Definizione 3.5 *Una* valutazione booleana

$$I_{\mathcal{V}} : \textsc{Prop} \mapsto \{1, 0\}$$

è l'estensione a PROP *di un'assegnazione booleana, cioè*

$$\begin{aligned} I_{\mathcal{V}}(A) &= \mathcal{V}(A) \text{ se } A \in \mathcal{P}; \\ I_{\mathcal{V}}(\top) &= 1; \\ I_{\mathcal{V}}(\bot) &= 0; \\ I_{\mathcal{V}}(\neg A) &= \mathcal{O}p_{\neg}(I_{\mathcal{V}}(A)); \\ I_{\mathcal{V}}(A \circ B) &= \mathcal{O}p_{\circ}(I_{\mathcal{V}}(A), I_{\mathcal{V}}(B)). \end{aligned}$$

dove $\circ \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$. Data una assegnazione booleana $\mathcal{V}$, l'esistenza e l'unicità della estensione $I_{\mathcal{V}}$ sono garantite dal teorema di ricorsione 1.9.
Il valore di verità di una formula A dipende dall'assegnazione booleana a tutti i simboli proposizionali, inclusi quelli che non occorrono in A. Tuttavia, per dare una valutazione booleana di A basta dire come vengono valutati (interpretati) i simboli proposizionali di A. Infatti, sia A una formula proposizionale e sia $simb(A)$ (si veda l'esercizio 58) l'insieme dei simboli proposizionali che occorrono in A:

Teorema 3.6 *Se $\mathcal{V}_1$ e $\mathcal{V}_2$ sono due assegnazioni che coincidono su $simb(A)$ allora $I_{\mathcal{V}_1}(A) = I_{\mathcal{V}_2}(A)$.*

Dimostrazione Procediamo per induzione strutturale.
(*Passo Base*) Osserviamo che l'asserto è vero per le formule $\bot$ e $\top$ perché non contengono simboli proposizionali. Se $A \in \mathcal{P}$, allora abbiamo $\mathcal{V}_1(A) = \mathcal{V}_2(A)$ e, per definizione, $I_{\mathcal{V}_1}(A) = I_{\mathcal{V}_2}(A)$. Pertanto l'asserto vale per i simboli proposizionali.
(*Passo Induttivo*) Sia $A = (\neg B)$. Abbiamo $I_{\mathcal{V}_1}(\neg B) = \mathcal{O}p_{\neg}(I_{\mathcal{V}_1}(B))$ e $I_{\mathcal{V}_2}(\neg B) = \mathcal{O}p_{\neg}(I_{\mathcal{V}_2}(B))$ e, dunque, poiché per ipotesi induttiva $I_{\mathcal{V}_1}(B) = I_{\mathcal{V}_2}(B)$, segue che $I_{\mathcal{V}_1}(A) = I_{\mathcal{V}_2}(A)$. Analogamente, sia $A = B \circ C$ con $\circ \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$; abbiamo $I_{\mathcal{V}_1}(B \circ C) = \mathcal{O}p_{\circ}(I_{\mathcal{V}_1}(B), I_{\mathcal{V}_1}(C))$ e $I_{\mathcal{V}_2}(B \circ C) = \mathcal{O}p_{\circ}(I_{\mathcal{V}_2}(B), I_{\mathcal{V}_2}(C))$ e, siccome per ipotesi induttiva, $I_{\mathcal{V}_1}(B) = I_{\mathcal{V}_2}(B)$ e $I_{\mathcal{V}_1}(C) = I_{\mathcal{V}_2}(C)$, segue che $I_{\mathcal{V}_1}(A) = I_{\mathcal{V}_2}(A)$. □

Esempio 28 *Consideriamo la formula $A \rightarrow ((B \wedge C) \vee (C \rightarrow \neg A))$ sia $\mathcal{V}(A) = \mathcal{V}(B) = 1$ e $\mathcal{V}(C) = 0$. Abbiamo:*

$$\begin{aligned} & I_{\mathcal{V}}(A \rightarrow ((B \wedge C) \vee (C \rightarrow \neg A))) = \\ = \; & \mathcal{O}p_{\rightarrow}(I_{\mathcal{V}}(A), I_{\mathcal{V}}((B \wedge C) \vee (C \rightarrow \neg A))) \\ = \; & \mathcal{O}p_{\rightarrow}(1, \mathcal{O}p_{\vee}(I_{\mathcal{V}}(B \wedge C), I_{\mathcal{V}}(C \rightarrow \neg A))) \end{aligned}$$

$$
\begin{aligned}
&= \mathcal{O}p_{\rightarrow}(1, \mathcal{O}p_{\vee}(\mathcal{O}p_{\wedge}(I_{\mathcal{V}}(B), I_{\mathcal{V}}(C)), \mathcal{O}p_{\rightarrow}(I_{\mathcal{V}}(C), \mathcal{O}p_{\neg}(I_{\mathcal{V}}(A))))) \\
&= \mathcal{O}p_{\rightarrow}(1, \mathcal{O}p_{\vee}(\mathcal{O}p_{\wedge}(1,0), \mathcal{O}p_{\rightarrow}(0,0))) \\
&= \mathcal{O}p_{\rightarrow}(1, \mathcal{O}p_{\vee}(0,1)) \\
&= \mathcal{O}p_{\rightarrow}(1,1) \\
&= 1
\end{aligned}
$$

ovvero: $I_{\mathcal{V}}$ esiste, univocamente estende $\mathcal{V}$ e il valore di verità della formula è determinato dal valore che $\mathcal{V}$ assume sui simboli proposizionali A, B e C che occorrono in essa.

Definizione 3.7 *Una formula proposizionale A è* soddisfatta *da una valutazione booleana $I_{\mathcal{V}}$ se $I_{\mathcal{V}}(A) = 1$.*

Definizione 3.8 *Una formula proposizionale A è* soddisfacibile *se è soddisfatta da una qualche valutazione booleana $I_{\mathcal{V}}$.*

Definizione 3.9 *Una formula proposizionale A è una* tautologia *se è soddisfatta da ogni valutazione booleana $I_{\mathcal{V}}$.*

Definizione 3.10 *Una formula proposizionale A è una* contraddizione *non è soddisfatta da nessuna valutazione booleana $I_{\mathcal{V}}$.*

Dal teorema 3.6 segue che, per determinare se una formula proposizionale A è soddisfacibile, tautologica o contraddittoria, basta costruire una tabella di valori di verità in cui compaiono $n+1$ colonne, dove n è il numero di simboli proposizionali distinti $A_1, A_2, \cdots A_n$ che compaiono in A e nella $n+1$-esima colonna viene riportato il corrispondente valore $I_{\mathcal{V}}(A)$. La tabella ha 2^n righe, tante sono le possibili assegnazioni distinte di valori di verità ai simboli di A.

A_1	A_2	$\ldots$	A_n	$I_{\mathcal{V}}(A)$
1	1	$\ldots$	1	$I_{\mathcal{V}_1}(A)$
1	1	$\cdots$	0	$I_{\mathcal{V}_2}(A)$
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
0	0	$\cdots$	0	$I_{\mathcal{V}_{2^n}}(A)$

A è una tautologia sse l'ultima colonna contiene solo 1, è soddisfacibile sse essa contiene almeno un 1, è contraddittoria sse essa contiene tutti 0.

Teorema 3.11 *Una formula A è una tautologia sse $\neg A$ è una contraddizione.*

Dimostrazione Lasciata al lettore (si veda l'esercizio 60). □

Definizione 3.12 *Diciamo che* A implica logicamente B *sse ogniqualvolta* $I_{\mathcal{V}}(A) = 1$ *anche* $I_{\mathcal{V}}(B) = 1$.

Definizione 3.13 *Diciamo che due proposizioni* A *e* B *sono* logicamente equivalenti *o* tautologicamente equivalenti *sse* $I_{\mathcal{V}}(A) = I_{\mathcal{V}}(B)$ *per ogni valutazione booleana* $I_{\mathcal{V}}$. *In tal caso scriviamo* $A \equiv B$.

Diamo qui di seguito una lista di formule tautologicamente equivalenti. Il lettore può verificare l'equivalenza, quale facile esercizio (si veda l'esercizio 61).

1. Idempotenza:

$$\begin{aligned} A \wedge A &\equiv A \\ A \vee A &\equiv A \end{aligned}$$

2. Associatività:

$$\begin{aligned} A \wedge (B \wedge C) &\equiv (A \wedge B) \wedge C \\ A \vee (B \vee C) &\equiv (A \vee B) \vee C \\ A \leftrightarrow (B \leftrightarrow C) &\equiv (A \leftrightarrow B) \leftrightarrow C \end{aligned}$$

3. Commutatività:

$$\begin{aligned} A \wedge B &\equiv B \wedge A \\ A \vee B &\equiv B \vee A \\ A \leftrightarrow B &\equiv B \leftrightarrow A \end{aligned}$$

4. Distributività:

$$\begin{aligned} A \wedge (B \vee C) &\equiv (A \wedge B) \vee (A \wedge C) \\ A \vee (B \wedge C) &\equiv (A \vee B) \wedge (A \vee C) \end{aligned}$$

5. Assorbimento:

$$\begin{aligned} A \wedge (A \vee B) &\equiv A \\ A \vee (A \wedge B) &\equiv A \end{aligned}$$

6. Doppia negazione:

$$\neg\neg A \equiv A$$

7. Leggi di De Morgan:

$$\begin{aligned} \neg(A \wedge B) &\equiv \neg A \vee \neg B \\ \neg(A \vee B) &\equiv \neg A \wedge \neg B \end{aligned}$$

8. Terzo escluso:[1]

$$A \vee \neg A \equiv \top$$

9. Contrapposizione:

$$A \rightarrow B \equiv \neg B \rightarrow \neg A$$

10. Contraddizione:

$$A \wedge \neg A \equiv \bot.$$

Intuitivamente ci si aspetta che, se una proposizione A è tautologicamente equivalente a una proposizione B, sostituendo A al posto di B all'interno di una formula C il valore di verità di C non cambi.

Cerchiamo ora di rendere più precisa questa nozione. Indichiamo con $F[p]$ una formula proposizionale che può contenere delle occorrenze del simbolo p (detto *metavariabile* o *parametro* proposizionale, cioè p sta a indicare un atomo). Con la notazione $F[X/p]$ indicheremo la formula $F[X]$ in cui tutte le occorrenze di p sono state uniformemente e simultaneamente sostituite con occorrenze di una formula X. Chiameremo questa operazione *sostituzione uniforme*.

Teorema 3.14 [RIMPIAZZAMENTO] *Siano $F[p]$, X e Y formule proposizionali e sia $I_{\mathcal{V}}$ una valutazione booleana. Se $I_{\mathcal{V}}(X) = I_{\mathcal{V}}(Y)$ allora*
$I_{\mathcal{V}}(F[X/p]) = I_{\mathcal{V}}(F[Y/p])$.

Dimostrazione Per induzione strutturale. Osserviamo, innanzi tutto che se p non occorre in F, nessuna sostituzione è stata fatta, e quindi l'asserto è banalmente verificato.
(*Passo Base*) Supponiamo che $F[p] = p$ allora $F[X/p] = X$ e $F[Y/p] = Y$; per ipotesi $I_{\mathcal{V}}(X) = I_{\mathcal{V}}(Y)$ e dunque $I_{\mathcal{V}}(F[X/p]) = I_{\mathcal{V}}(X) = I_{\mathcal{V}}(Y) = I_{\mathcal{V}}(F[Y/p])$.
(*Passo Induttivo*) Sia $F[p] = A[p] \wedge B[p]$. Per ipotesi abbiamo che $I_{\mathcal{V}}(X) = I_{\mathcal{V}}(Y)$; osserviamo che, per definizione di valutazione booleana:

$$I_{\mathcal{V}}(F[X/p]) = \mathcal{O}p_{\wedge}(I_{\mathcal{V}}(A[X/p]), I_{\mathcal{V}}(B[X/p])).$$

[1] Il fatto che $A \vee \neg A$ sia sempre vero sembra del tutto intuitivo. In effetti questa è una caratteristica della logica classica; in altre logiche, per esempio nella logica intuizionista, questo fatto non vale.

Per ipotesi induttiva: $I_{\mathcal{V}}(A[X/p]) = I_{\mathcal{V}}(A[Y/p])$ e $I_{\mathcal{V}}(B[X/p]) = I_{\mathcal{V}}(B[Y/p])$. Quindi:

$$\mathcal{O}p_{\wedge}(I_{\mathcal{V}}(A[X/p]), I_{\mathcal{V}}(B[X/p])) = \mathcal{O}p_{\wedge}(I_{\mathcal{V}}(A[Y/p]), I_{\mathcal{V}}(B[Y/p]))$$

Di nuovo, per definizione di valutazione booleana:

$$\mathcal{O}p_{\wedge}(I_{\mathcal{V}}(A[Y/p]), I_{\mathcal{V}}(B[Y/p])) = I_{\mathcal{V}}(F[Y/p]).$$

Dunque

$$I_{\mathcal{V}}(F[X/p]) = I_{\mathcal{V}}(F[Y/p]).$$

Il caso degli altri connettivi è analogo ed è lasciato al lettore (si veda l'esercizio 62). □

Teorema 3.15 *Se $X \equiv Y$ allora $F[X/p] \equiv F[Y/p]$.*

Dimostrazione Sia $X \equiv Y$ una equivalenza tautologica, quindi $I_{\mathcal{V}}(X) = I_{\mathcal{V}}(Y)$ per ogni valutazione booleana $I_{\mathcal{V}}$; per il teorema 3.14, $I_{\mathcal{V}}(F[X/p]) = I_{\mathcal{V}}(F[Y/p])$ per ogni valutazione booleana. Ne segue che $F[X/p] \equiv F[Y/p]$. □

Esempio 29 *Si consideri l'equivalenza tautologica*

$$(A \to B) \equiv (\neg A \vee B).$$

Sia $F[p] = (p \to (C \vee D))$. Si può facilmente verificare che

$$((A \to B) \to (C \vee D)) \equiv ((\neg A \vee B) \to (C \vee D)).$$

Come abbiamo visto nel capitolo 2, possiamo fornire una nozione di interpretazione basata sulla relazione[2] $\models$. Sia $\mathcal{M}$ un insieme di simboli proposizionali, definiamo $\models \subseteq (\mathcal{M} \times \mathcal{L})$ ricorsivamente come segue:

1. $\mathcal{M} \models A$ sse $A \in \mathcal{M}$;
2. $\mathcal{M} \models \top$ e $\mathcal{M} \not\models \bot$;
3. $\mathcal{M} \models \neg A$ sse non $(\mathcal{M} \models A)$ sse $\mathcal{M} \not\models A$;
4. $\mathcal{M} \models A \wedge B$ sse $\mathcal{M} \models A$ e $\mathcal{M} \models B$;
5. $\mathcal{M} \models A \vee B$ sse $\mathcal{M} \models A$ oppure $\mathcal{M} \models B$;
6. $\mathcal{M} \models A \to B$ sse $\mathcal{M} \not\models A$ oppure $\mathcal{M} \models B$;
7. $\mathcal{M} \models A \leftrightarrow B$ sse $\mathcal{M} \models A$ e $\mathcal{M} \models B$, oppure $\mathcal{M} \not\models A$ e $\mathcal{M} \not\models B$.

[2] Come già detto nel capitolo 2, $\mathcal{M} \models A$ si legge "$\mathcal{M}$ modella A".

La relazione con la valutazione booleana è data dal fatto che l'insieme dei simboli proposizionali che appartengono a $\mathcal{M}$ hanno valutazione booleana 1, e gli altri hanno valutazione booleana 0, ovvero:

$$p \in \mathcal{M} \text{ sse } \mathcal{V}(p) = 1.$$

Dunque il passo base della definizione ricorsiva di $\models$ coincide con la definizione di assegnazione booleana $\mathcal{V}$. Il passo ricorsivo, sui connettivi $\neg$ e $\wedge$, in realtà, sfrutta una nozione meta-logica come "non" ed "e". Tuttavia la definizione è giustificata dal fatto che si può basare su $I_{\mathcal{V}}$, che estende unicamente $\mathcal{V}$.

Osserviamo che, nella definizione di $\models$, stiamo sottintendendo il linguaggio proposizionale $\mathcal{L}$. Nel caso in cui non sia chiaro dal contesto a quale linguaggio ci si riferisce, allora si usa indicare $\models_{\mathcal{L}}$.

Definizione 3.16 *Sia A una formula, se $\mathcal{M} \models A$ diciamo che $\mathcal{M}$ è un* modello *di A, ovvero che $\mathcal{M}$* rende vera *A.*

Definizione 3.17 *Se $\mathcal{M}$ rende vere tutte le formule di un insieme Γ, cioè se $\mathcal{M} \models A$, per ogni formula A in Γ, diciamo che $\mathcal{M}$ è* un modello *per Γ e indichiamo questo con $\mathcal{M} \models \Gamma$.*

Se A è una tautologia, possiamo scrivere $\models A$, in quanto A è vera in ogni modello, compreso quello vuoto.

Definizione 3.18 *Se $\mathcal{M} \models A$ per qualche $\mathcal{M}$, allora diciamo che A è soddisfacibile.*

Definizione 3.19 *Se per nessun insieme di simboli proposizionali $\mathcal{M}$ è verificato che $\mathcal{M} \models A$ allora diciamo che A è* insoddisfacibile.

Quindi una formula è insoddisfacibile sse per essa non esiste un modello.

Esempio 30

1. *La formula $A \wedge B$ ha $\{A, B\}$ come modello. Dunque la formula è soddisfacibile.*
2. *La formula $A \wedge \neg B$ ha $\{A\}$ come modello.*
3. *La formula $A \wedge \neg A$ non ha alcun modello. Dunque la formula non è soddisfacibile.*
4. *La formula $A \rightarrow B$ ha $\{\}$, $\{B\}$ e $\{A, B\}$ come modelli; mentre $\{A\}$ non è un modello.*
5. *La formula $A \vee B$ ha $\{A\}$, $\{B\}$ e $\{A, B\}$ come modelli.*
6. *Sia $A \rightarrow ((B \wedge C) \vee (C \rightarrow \neg A))$, la formula dell'esempio 28. $\mathcal{M} = \{A, B\}$ è un modello per essa. Il lettore verifichi se la formula ha o no altri modelli.*

Il simbolo $\models$ è giustificato quando si considerano le nozioni di implicazione tautologica e di equivalenza tautologica, perché permettono una semplificazione notazionale.

Infatti se A *implica tautologicamente* B scriviamo $\models A \rightarrow B$. Dato un insieme di proposizioni Γ e una proposizione A, se Γ *implica logicamente* A scriviamo $\Gamma \models A$. Infine, se A è *tautologicamente equivalente a* B, cioè se $A \equiv B$, scriviamo $\models A \leftrightarrow B$.

Il seguente teorema lega le nozioni di implicazione logica e di insoddisfacibilità:

Teorema 3.20

$$\Gamma \models A \textit{ sse } \Gamma \cup \{\neg A\} \textit{ è insoddisfacibile.}$$

Dimostrazione
($\Rightarrow$) Supponiamo $\Gamma \models A$, ma $\Gamma \cup \{\neg A\}$ soddisfacibile. Allora esiste un modello $\mathcal{M}$ tale che $\mathcal{M} \models \Gamma$ e $\mathcal{M} \models \neg A$, contraddizione con la definizione di implicazione logica.
($\Leftarrow$) Supponiamo che $\Gamma \cup \{\neg A\}$ sia insoddisfacibile, consideriamo le seguenti trasformazioni:

1. non esiste un modello $\mathcal{M}$ tale che $\mathcal{M}$ verifica entrambi gli asserti $\mathcal{M} \models \Gamma$ e $\mathcal{M} \models \neg A$;
2. per ogni modello $\mathcal{M}$, $\mathcal{M}$ non verifica entrambi gli asserti $\mathcal{M} \models \Gamma$ e $\mathcal{M} \models \neg A$;
3. per ogni modello $\mathcal{M}$, $\mathcal{M} \not\models \Gamma$ oppure $\mathcal{M} \not\models \neg A$;
4. per ogni modello $\mathcal{M}$, $\mathcal{M} \models \Gamma$ implica $\mathcal{M} \not\models \neg A$;
5. $\mathcal{M} \not\models \neg A$ sse $\mathcal{M} \models A$;
6. per ogni modello $\mathcal{M}$, $\mathcal{M} \models \Gamma$ implica $\mathcal{M} \models A$;
7. $\Gamma \models A$.

È facile verificare che 1 implica 2, 2 implica 3, ... e 6 implica 7. Infatti le trasformazioni sono manipolazioni, a livello logico, di costrutti meta-logici come "per tutti", "esiste", "e", "oppure", "non", "implica", eccetera. □

La proprietà enunciata nel teorema 3.20 è quella che giustifica le dimostrazioni per assurdo: se si vuole provare che A segue da Γ basta assumere $\neg A$ e provare che questo contraddice Γ.

Dall'esempio 30 segue quanto abbiamo già notato per le valutazioni booleane: quando consideriamo i modelli di una formula basta prendere in considerazione solo quelli in cui compaiono i simboli proposizionali della formula. È chiaro che se in una formula A occorrono n simboli proposizionali distinti, ovvero $|simb(A)| = n$, allora il numero degli insiemi di simboli che ci interessa verificare per conoscere il valore di verità di A è al più 2^n. Tuttavia, cosa succede se abbiamo un insieme infinito Γ di proposizioni e vogliamo sapere se $\Gamma \models A$?

Il teorema di compattezza della soddisfacibilità (3.24) fornisce una risposta a questa domanda. Introduciamo prima alcune definizioni e un lemma che saranno utilizzati nella sua dimostrazione.

Definizione 3.21 *Un insieme di formule* Γ *si dice* finitamente soddisfacibile *sse ogni suo sottoinsieme finito è soddisfacibile.*

Definizione 3.22 *Un insieme di formule* Γ *si dice* massimale *sse per ogni* A *di* $\mathcal{L}$ *si ha* $A \in \Gamma$ *oppure* $\neg A \in \Gamma$.

Lemma 3.23 *Sia* Γ *un insieme di formule. Se ogni sottoinsieme finito di* Γ *è soddisfacibile, allora per ogni formula* A*, ogni sottoinsieme finito di* $\Gamma \cup \{A\}$ *o di* $\Gamma \cup \{\neg A\}$ *è soddisfacibile.*

Dimostrazione Lasciata al lettore (si veda l'esercizio 63). □

Teorema 3.24 [COMPATTEZZA DELLA SODDISFACIBILITÀ] *Un insieme di proposizioni* Γ *è soddisfacibile sse è finitamente soddisfacibile.*

Dimostrazione Osserviamo innanzi tutto che se Γ è finito la tesi è dimostrata. Ci occupiamo quindi di Γ infiniti.
($\Rightarrow$) Se Γ è soddisfacibile, allora è finitamente soddisfacibile. Supponiamo Γ soddisfacibile e sia $\mathcal{M}_0$ un modello che lo soddisfa. Sia Γ_0 un generico sottoinsieme finito di Γ. Per ogni formula $A \in \Gamma_0 \subset \Gamma$ abbiamo che $\mathcal{M}_0 \models A$. Quindi $\mathcal{M}_0 \models \Gamma$.
($\Leftarrow$) Se un insieme di formule Γ infinito è finitamente soddisfacibile, allora è soddisfacibile. La dimostrazione consiste di due parti. Nella prima dimostriamo che un insieme finitamente soddisfacibile Γ si può estendere a un insieme massimale Δ finitamente soddisfacibile. Nella seconda parte costruiremo un modello per Δ. Siccome abbiamo costruito Δ estendendo Γ, tale modello sarà anche un modello per Γ, quindi la tesi sarà dimostrata.

Cominciamo con l'enumerare tutte le formule del linguaggio: $A_0, A_1, A_2, \ldots$. Questa enumerazione è possibile per il teorema 1.3. Definiamo per ricorsione:

$$\begin{aligned} \Delta_0 &= \Gamma \\ \Delta_n &= \begin{cases} \Delta_{n-1} \cup A_{n-1}, \text{ se } \Delta_{n-1} \cup A_{n-1} \text{ è finitamente soddisfacibile;} \\ \Delta_{n-1} \cup \neg A_{n-1}, \text{ altrimenti.} \end{cases} \end{aligned}$$

Sia $\Delta = \bigcup_n \Delta_n$. Osserviamo innanzi tutto che $\Gamma \subseteq \Delta$, per costruzione. Osserviamo inoltre che Δ è massimale, cioè per ogni formula A, A o la sua negazione appartengono a Δ. Infatti, siccome la enumerazione delle formule è completa, per costruzione dei Δ_i, presa una qualunque A, esisterà un n per cui $A = A_n$, quindi $A \in \Delta_n \subseteq \Delta$ oppure $\neg A \in \Delta_n \subseteq \Delta$. Osserviamo anche che ciascun Δ_n è finitamente soddisfacibile (per induzione: $\Delta_0 = \Gamma$ è finitamente soddisfacibile; per il passo induttivo si usa il lemma 3.23). Quindi, Δ è finitamente soddisfacibile, in quanto ogni sottoinsieme finito di Δ è contenuto in qualche Δ_n.

Dobbiamo ora mostrare che Δ è soddisfacibile. Per far questo costruiamo un modello per esso. Fissiamo, per ogni simbolo proposizionale P che occorre in Δ:

$$P \in \Delta \text{ sse } P \in \mathcal{M}.$$

Dimostriamo ora, per induzione strutturale sulle formule, che per ogni formula A:

$$A \in \Delta \text{ sse } \mathcal{M} \models A.$$

(*Passo Base*) Per costruzione di $\mathcal{M}$.
(*Passo Induttivo*) Sia $A = \neg B$. $\mathcal{M} \models \neg B$ equivale a $\mathcal{M} \not\models B$; per ipotesi induttiva, $\mathcal{M} \not\models B$ sse $B \notin \Delta$. Siccome Δ è massimale, da $B \notin \Delta$ segue $\neg B \in \Delta$. Pertanto $\mathcal{M} \models \neg B$ sse $\neg B \in \Delta$, che è la tesi.
Sia $A = B \wedge C$. $\mathcal{M} \models B \wedge C$ equivale a $\mathcal{M} \models B$ e $\mathcal{M} \models C$; per ipotesi induttiva, abbiamo $\mathcal{M} \models B$ sse $B \in \Delta$ e $\mathcal{M} \models C$ sse $C \in \Delta$. Quindi $B, C \in \Delta$, ovvero $B \wedge C \in \Delta$, che è la tesi. Il caso degli altri connettivi binari è analogo ed è lasciato al lettore (si veda l'esercizio 64).

Abbiamo quindi mostrato che $\mathcal{M}$ è un modello per Δ; siccome $\Gamma \subseteq \Delta$, abbiamo che $\mathcal{M} \models \Gamma$, ovvero Γ è soddisfacibile. □

Il teorema di compattezza può essere enunciato in modo equivalente come segue:

Teorema 3.25 *Un insieme Γ di formule è insoddisfacibile sse esiste un sottoinsieme finito $\Delta \subseteq \Gamma$ che è insoddisfacibile.*

La dimostrazione che questo enunciato è equivalente al precedente è lasciata al lettore (si veda l'esecizio 65). □

Dal teorema di compattezza seguono delle proprietà interessanti, per esempio il seguente corollario.

Corollario 3.26 *Se $\Gamma \models A$ allora esiste un sottoinsieme finito Γ_0 di Γ, tale che $\Gamma_0 \models A$.*

Dimostrazione Per contraddizione. Supponiamo che per ogni Γ_0 finito, $\Gamma_0 \not\models A$, e consideriamo le seguenti trasformazioni:

1. $\Gamma_0 \not\models A$, per ogni Γ_0 finito, ipotesi;
2. $\Gamma_0 \cup \{\neg A\}$ è soddisfacibile per ogni Γ_0 finito (da 1, per il teorema 3.20);
3. $\Gamma \cup \{\neg A\}$ è finitamente soddisfacibile (da 2, per definizione di insieme finitamente soddisfacibile 3.21);
4. $\Gamma \cup \{\neg A\}$ è soddisfacibile (da 3, per il teorema di compattezza 3.24);
5. $\Gamma \not\models A$ (da 4, per il teorema 3.20).

Contraddizione. □

Esempio 31 *Verifichiamo che $A \wedge B \models B$ e che $A \wedge B \cup \{\neg B\}$ è insoddisfacibile.*

Sia $\mathcal{M} = \{A, B\}$, ogni modello $\mathcal{M}'$ che contiene $\mathcal{M}$ è un modello di $A \wedge B$ e, ovviamente, un modello di B. Si osservi che $\mathcal{M}^* = \{B\}$ è un modello di B, ma non è un modello di $A \wedge B$. Più in generale, nessun sottoinsieme proprio di $\mathcal{M}$ è un modello di $A \wedge B$.

$A \wedge B \cup \{\neg B\}$, può essere scritto come $A \wedge (B \wedge \neg B)$; è evidente che tale formula non ha modelli.

3.2.1 Esercizi

Esercizio 60 *Dimostrare il teorema 3.11.*

Esercizio 61 *Dimostrare le equivalenze tautologiche di pagina 67 e successiva.*

Esercizio 62 *Dimostrare il caso* $F[p] = \neg A[p]$ *e gli altri casi non dimostrati nel teorema 3.14.*

Esercizio 63 *Dimostrare il teorema 3.23.*

Esercizio 64 *Completare la dimostrazione del teorema 3.24.*

Esercizio 65 *Dimostrare che l'enunciato del teorema 3.24 è equivalente a quello del teorema 3.25.*

Esercizio 66 *Individuare quali delle seguenti formule è una tautologia:*

1. $A \rightarrow (B \rightarrow A)$;
2. $(A \rightarrow B) \rightarrow ((A \rightarrow \neg B) \rightarrow \neg A)$;
3. $A \rightarrow (\neg A \rightarrow B)$;
4. $A \rightarrow (A \rightarrow \bot))$;
5. $(\bot \rightarrow A)$.

Esercizio 67 *Dimostrare che la seguente formula, costruita su un numero finito di simboli* $A_1, A_2, \ldots, A_n$ *è una tautologia se e solo se ogni simbolo proposizionale* A_i *compare un numero pari di volte:*

$$A_1 \leftrightarrow A_2 \leftrightarrow A_1 \leftrightarrow A_n \leftrightarrow A_2 \ldots \leftrightarrow A_1.$$

Esercizio 68 *Siano* $A_1, A_2, \ldots, A_n$, *con* n *numero finito pari, formule del calcolo proposizionale e sia* P *un simbolo proposizionale. Dimostrare che la formula:*

$$A_1 \leftrightarrow A_2 \leftrightarrow \ldots \leftrightarrow A_n$$

è soddisfacibile se lo è il seguente insieme di formule.

$$\{P \leftrightarrow A_1, P \leftrightarrow A_2, \ldots, P \leftrightarrow A_n\}.$$

Il viceversa è falso; fornire un controesempio.

Esercizio 69 *Definire una procedura che assegnando arbitrariamente un valore di verità alle foglie dell'albero sintattico associato a una formula proposizionale (si veda l'esercizio 59), stampi il valore booleano della formula che etichetta la radice.*

Esercizio 70 *Considerare la seguente tautologia* $F = (((B \to A) \to B) \to B)$. *Dimostrare che* $\models F$ *sse* $I_{\mathcal{V}}(F) = 1$ *per ogni valutazione booleana* $I_{\mathcal{V}}$.

Esercizio 71 *Considerare la formula e il modello* $\mathcal{M}$ *dell'esempio 30, parte 1. Considerare un qualunque simbolo proposizionale di* $\mathcal{L}$. *Cosa succede se se ne modifica l'interpretazione?*

Esercizio 72 *Dimostrare le seguenti equivalenze tautologiche:*

1. $(A \leftrightarrow B) \equiv (A \to B) \wedge (B \to A)$;
2. $(A \to B) \equiv \neg A \vee B$;
3. $\neg \top \equiv \bot$.

Esercizio 73 *Dimostrare che* $A \models B$ *sse* $\models A \to B$.

Esercizio 74 *Dimostrare che* $A_1, \ldots, A_n \models B$ *sse* $A_1, \ldots, A_{n-1} \models A_n \to B$, *per ogni* $n > 0$.

Esercizio 75 *Sia* $\mathcal{L}$ *un linguaggio proposizionale costruito su un alfabeto di n simboli, con n finito. Sia* $\mathcal{M}od = \{\mathcal{M} | \mathcal{M} \models A, A \in \mathcal{L}\}$, *cioè* $\mathcal{M}od$ *è l'insieme di tutti i modelli di* $\mathcal{L}$. *Dimostrare che la cardinalità di* $\mathcal{M}od$ *è al massimo* 2^n.

Esercizio 76 *Un enunciato del calcolo proposizionale è detto* positivo *se è un atomo oppure se è costruito dagli atomi usando solo i connettivi* $\wedge$ *e* $\vee$. *Sia* Γ *un insieme finito e soddisfacibile di formule di un linguaggio proposizionale* $\mathcal{L}$.
Sia $\Delta = \{A | \Gamma \models A, A$ *è un enunciato positivo*$\}$. *Dimostrare che* $\mathcal{N}$ *è un modello di* Δ *sse esiste un* $\mathcal{M}$ *tale che* $\mathcal{M}$ *è un modello di* Γ *e* $\mathcal{N} \subseteq \mathcal{M}$.

3.3 Decidibilità

La logica proposizionale è decidibile. In effetti, la logica proposizionale si chiama calcolo delle proposizioni perché esiste una *procedura effettiva* che stabilisce se una proposizione A è una tautologia o meno. Nel caso del calcolo proposizionale possiamo sempre essere in grado di calcolare le formule vere del linguaggio. In verità, come vedremo nel caso della logica del primo ordine, la dizione calcolo si usa anche qualora si abbia una logica semidecidibile.

Per verificare la decidibilità del calcolo proposizionale siamo interessati a decidere se una formula A del calcolo proposizionale è una tautologia o meno.

Un altro interessante problema di decisione per il calcolo proposizionale consiste nello stabilire se una formula è soddisfacibile o meno (questo problema è di solito indicato con SAT (si veda 39).

Abbiamo visto che per decidere se A è una tautologia bisogna verificare le tabelle di verità per ogni assegnazione ai simboli proposizionali che occorrono in A, pertanto se in A occorrono n simboli proposizionali, sarà sufficiente verificare 2^n casi.

Nel seguito usiamo *tabelle di verità ridotte*. Si chiama ridotta una tabella di verità in cui le colonne vengono indiciate con tutti i simboli e connettivi che occorrono nella formula, i valori di verità dei simboli proposizionali vengono riportati sotto ogni occorrenza del simbolo nella formula; il valore di verità risultante per ciascuna sottoformula viene riportato sotto il connettivo principale della sottoformula stessa. In questo modo il valore di verità dell'intera formula sarà leggibile nella colonna sottostante al suo connettivo principale.

Esempio 32 *Scriviamo la tabella di verità ridotta per* $P \to (Q \to P)$

P	$\to$	Q	$\to$	P
1	1	1	1	1
1	1	0	1	1
0	1	1	0	0
0	1	0	1	0

Il valore della quarta colonna è $I_{\mathcal{V}}(Q \to P)$ *secondo le assegnazioni booleane a* P *e* Q*. Il valore della seconda colonna, ovvero della colonna del connettivo principale, è* $I_{\mathcal{V}}(P \to (Q \to P))$*.*

Lemma 3.27 *Se* Γ *è effettivamente enumerabile, allora l'insieme delle conseguenze tautologiche di* Γ*, ovvero l'insieme* $\{A \mid \Gamma \models A\}$ *è effettivamente enumerabile.*

Dimostrazione Dobbiamo mostrare che, data una formula A, se $\Gamma \models A$ allora esiste una procedura che risponde SÌ. Consideriamo una enumerazione $\Gamma_0, \Gamma_1, \Gamma_2, \ldots$ dei sottoinsiemi finiti di Γ. Data una formula A, consideriamo la lista dei sottoinsiemi finiti di Γ nel seguente modo:

$$\begin{aligned} &\models A, \\ \Gamma_0 &\models A, \\ \Gamma_1 &\models A, \\ &\ldots \end{aligned}$$

Per il corollario 3.26 se $\Gamma \models A$ esiste un Γ_i finito tale che $\Gamma_i \models A$, e dunque tale Γ_i comparirà nella enumerazione. □

Il problema della verifica se una formula del calcolo proposizionale è o meno una tautologia, come pure il problema SAT, sono chiaramente esponenziali nella dimensione della formula. Consideriamo come dimensione di una formula il numero di simboli di proposizione distinti che in essa compaiono. Come già detto, data una formula A con n simboli proposizionali distinti, per verificare se la formula è una tautologia o se è soddisfacibile basta costruire una tabella di verità per A, che conterrà 2^n righe.

Ci si può chiedere se si può fare di meglio, cioè se sia possibile trovare algoritmi polinomiali per risolvere il problema di decisione delle tautologie e della soddisfacibilità del calcolo proposizionale.

SAT non è ancora stato dimostrato intrinsecamente esponenziale, anche se esiste una certa evidenza in questo senso: se SAT fosse polinomiale tutta una serie di problemi a esso riducibili, per i quali non è stata dimostrata la polinomialità, sarebbe polinomiale, in quanto, come già visto nel teorema 1.31, SAT è $\mathcal{NP}$-completo.

Per quanto riguarda invece la verifica della tautologicità delle formule del calcolo proposizionale, si fa notare che questo problema è il complementare di SAT, infatti una formula è una tautologia sse la sua negata è non soddisfacibile. Esso verrà indicato con $\overline{SAT}$. Il problema di decisione per $\overline{SAT}$ è quindi co-$\mathcal{NP}$.

3.3.1 Esercizi

Esercizio 77 *Dimostrare se l'insieme delle formule proposizionali* PROP *è decidibile o meno.*

Esercizio 78 *Dimostrare se l'insieme delle tautologie è decidibile o meno.*

Esercizio 79 *Dimostrare se l'insieme delle formule soddisfacibili è decidibile o meno.*

Esercizio 80 *Sia* Γ *un insieme tale che per ogni* A *si verifica che* $\Gamma \models A$ *oppure* $\Gamma \models \neg A$*, dimostrare che* Γ *è decidibile.*

3.4 Completezza di insiemi di connettivi

I connettivi definiscono delle funzioni da $\{0,1\}$ (nel caso del connettivo unario $\neg$) oppure $\{0,1\} \times \{0,1\}$ (nel caso dei connettivi binari $\vee, \wedge, \ldots$) in $\{0,1\}$. Ciascuna di queste funzioni, dette *funzioni booleane*, è definita attraverso la relativa tabella dei valori di verità oppure, equivalentemente, attraverso le funzioni $\mathcal{O}p_\neg$ e $\mathcal{O}p_\circ$, $\circ \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$.

Più in generale, ogni formula proposizionale A contenente simboli proposizionali distinti $A_1, A_2, \ldots, A_n$ definisce una funzione booleana da $\{0,1\}^n$ in $\{0,1\}$.

Definizione 3.28 *Sia* A *una formula proposizionale contenente esattamente* n *atomi distinti* $A_1, A_2, \ldots, A_n$*; la funzione* $f_A : \{0,1\}^n \mapsto \{0,1\}$ *tale che* $f_A(v_1, v_2, \ldots, v_n) = I_{\mathcal{V}}(A)$*, dove* $\mathcal{V}$ *è l'interpretazione per cui* $\mathcal{V}(A_i) = v_i$ *per ogni* $i = 1, 2, \ldots, n$ *è detta la* funzione di verità *associata ad* A*.*

Quindi, ogni proposizione del calcolo proposizionale definisce una funzione n-aria (possiamo anche chiamarla connettivo n-ario), dove n è il numero degli atomi distinti che in essa compaiono. Si può facilmente verificare che, per ogni n esistono 2^{2^n} funzioni booleane distinte (cioè tante quanti sono i sottoinsiemi di $\{0,1\}^n$). Quindi, nel caso di $n = 2$ esistono 16 connettivi distinti. Noi ne abbiamo introdotti 4, e – come alcune equivalenze logiche introdotte precedentemente hanno mostrato – essi non sono indipendenti, nel senso che alcuni sono esprimibili in termini di altri.

Definizione 3.29 *Dato un insieme di connettivi $\mathcal{C}$ e un connettivo $c \notin \mathcal{C}$ per cui si abbia una funzione di verità $f_c = \mathcal{O}p_c$, si dice che c* si deriva da *(oppure si* definisce in termini dei*) connettivi di $\mathcal{C}$ se esiste una formula proposizionale F costruita usando solo i connettivi di $\mathcal{C}$ tale che $f_c \equiv f_F$.*

Esempio 33 *Il connettivo $\wedge$ si può definire in termini di $\{\neg, \vee\}$ nel seguente modo: $(A \wedge B) \equiv \neg(\neg A \vee \neg B)$.*

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

Esempio 34 *La formula $F = (A \wedge \neg B) \vee (\neg A \wedge B)$ definisce la seguente funzione di verità:*

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

Il connettivo binario così definito si chiama or esclusivo *perché corrisponde all'"aut" latino (esso è vero sse uno solo dei suoi argomenti è vero). In genere esso viene indicato con $\underline{\vee}$ e detto* xor. *La precedente costruzione mostra che il connettivo $\underline{\vee}$ è derivabile dall'insieme di connettivi $\{\neg, \wedge, \vee\}$.*

Teorema 3.30

$$(A \to B) \equiv (\neg A \vee B)$$

$$
\begin{aligned}
(A \vee B) &\equiv (\neg A \to B) \\
(A \vee B) &\equiv \neg(\neg A \wedge \neg B) \\
(A \wedge B) &\equiv \neg(\neg A \vee \neg B) \\
(A \wedge B) &\equiv (((A \to \bot) \to \bot) \to (B \to \bot)) \to \bot \\
\neg A &\equiv A \to \bot \\
\bot &\equiv A \wedge \neg A \\
(A \leftrightarrow B) &\equiv (A \to B) \wedge (B \to A)
\end{aligned}
$$

Dimostrazione Lasciata al lettore (si veda l'esercizio 81). □

Si noti che la costante proposizionale $\bot$ (così come la costante proposizionale $\top$) può essere considerata come un connettivo 0-ario.

Un insieme di connettivi logici è completo se mediante i suoi connettivi si può esprimere qualunque funzione booleana. Più formalmente:

Definizione 3.31 *Un insieme di connettivi logici $\mathcal{C}$ si dice* completo *sse, data una qualunque $f : \{0,1\}^n \mapsto \{0,1\}$ esiste una formula proposizionale A costruita mediante i connettivi dell'insieme $\mathcal{C}$ tale che $f \equiv f_A$.*

In particolare, un insieme di connettivi logici è completo se mediante i suoi connettivi si può esprimere qualunque altro connettivo.

È facile verificare (si veda l'esercizio 83) che l'insieme dei connettivi $\{\neg, \wedge, \vee\}$ è completo, così come lo sono: $\{\neg, \vee\}$ e anche $\{\neg, \wedge\}$.

Questo ci porta a concludere che nella trattazione della logica potremmo ridurci a considerare due soli connettivi. Si può in effetti anche dimostrare che uno solo connettivo binario può bastare per definire tutti gli altri; in particolare un connettivo a scelta tra il "nand" o il "nor" – definiti negli esercizi 87 e 88 – costituisce un insieme completo. Si può anche dimostrare (si veda l'esercizio 89) che nand e nor sono gli unici due connettivi binari completi.

Usare un minor numero di connettivi nelle formule porterebbe sicuramente a una maggiore stringatezza nelle dimostrazioni fatte per casi sui singoli connettivi, ma nuocerebbe grandemente alla leggibilità delle formule stesse. Le equivalenze mostrate nel teorema 3.30 ne forniscono evidenza. Abbiamo quindi preferito la leggibilità alla concisione.

3.4.1 Esercizi

Esercizio 81 *Dimostrare il teorema 3.30.*

Esercizio 82 *Mostrare come è possibile associare le operazioni insiemistiche ai connettivi. Dimostrare che le proprietà insiemistiche sono conservate.*

Esercizio 83 *Dimostrare che i seguenti insiemi di connettivi sono completi.*

1. $\{\neg, \wedge, \vee\}$;
2. $\{\neg, \vee\}$;
3. $\{\neg, \wedge\}$.

Esercizio 84 *Dimostrare che l'insieme di connettivi* $\{\rightarrow, \bot\}$ *è completo, sapendo che* $\{\neg, \wedge\}$ *lo è.*

Esercizio 85 *Dimostrare che l'insieme di connettivi* $\{\wedge, \leftrightarrow, \underline{\vee}\}$ *è completo, e che nessuno dei suoi sottoinsiemi propri lo è.*

Esercizio 86 *Stabilire se gli insiemi di connettivi* $\{\rightarrow, \wedge\}$ *e* $\{\vee, \wedge\}$ *sono o no completi.*

Esercizio 87 *Il connettivo* nand *è definito dalla seguente tabella di verità:*

A	B	A *nand* B
1	1	0
1	0	1
0	1	1
0	0	1

1. *Esprimerlo in funzione di* $\{\neg, \wedge\}$ *e di* $\{\neg, \vee\}$
2. *Dimostrare che esso è completo.*

Esercizio 88 *Il connettivo* nor *è definito dalla seguente tabella di verità:*

A	B	A *nor* B
1	1	0
1	0	0
0	1	0
0	0	1

1. *Esprimerlo in funzione di* $\{\neg, \wedge\}$ *e di* $\{\neg, \vee\}$
2. *Dimostrare che esso è completo.*

Esercizio 89 *Dimostrare che* nand *e* nor *sono gli unici due connettivi binari che da soli costituiscono un insieme completo.*

Esercizio 90 *Sia* $\bigtriangledown$ *il connettivo ternario espresso dalla seguente funzione di verità*

$$\bigtriangledown(A, B, C) = 1 \quad sse \quad max(\mathcal{V}(A), \mathcal{V}(B)) + \mathcal{V}(C) = 2$$

Esprimere $\bigtriangledown$ *in funzione di* $\{\neg, \wedge\}$.

3.5 Riepilogo

In questo capitolo abbiamo visto come le nozioni relative ai sistemi formali presentate nel capitolo precedente si calano nel caso di un semplice sistema formale, quello del calcolo proposizionale. In particolare, abbiamo introdotto:

1. il linguaggio, cioè l'insieme delle formule ben formate del calcolo proposizionale;
2. il sistema di valutazione e le tabelle dei valori di verità per i connettivi logici;
   (a) la nozione di formula tautologica, soddisfacibile, contraddittoria,
   (b) il teorema del rimpiazzamento,
   (c) la nozione di modello,
   (d) il teorema di compattezza della soddisfacibilità.
3. Abbiamo poi discusso la decidibiltà del calcolo proposizionale
4. e la completezza di insiemi di connettivi.

Capitolo 4
Sistemi deduttivi e deduzione automatica in logica proposizionale

Introduciamo, in questo capitolo, la nozione di *derivazione* di una formula proposizionale da un insieme di proposizioni o *teoria*. Per fare ciò occorre definire un sistema deduttivo, cioè un metodo di calcolo che manipoli esclusivamente proposizioni, che non necessiti di ricorrere ad altri aspetti della logica, come l'interpretazione delle formule. Dunque siamo interessati a definire un metodo di calcolo *indipendente* dalle funzioni di valutazione e dai modelli. Vedremo che si può stabilire, tramite un teorema di correttezza e completezza, una corrispondenza tra le formule che conseguono dalla teoria mediante tale metodo e le formule verificabili nei modelli della teoria.

In questo capitolo presenteremo vari sistemi deduttivi per il calcolo proposizionale: un sistema assiomatico, la deduzione naturale, il metodo dei tableau e la risoluzione. Il primo sistema si basa su un insieme di assiomi logici e un insieme di regole di inferenza; gli altri invece non fanno uso di assiomi logici, ma solo di regole di inferenza. Alcuni sono più adatti a una meccanizzazione, altri meno; questo aspetto verrà discusso nei rispettivi paragrafi. Tra i metodi presentati, solo la risoluzione richiede che le formule siano scritte in una forma normale. Tutti sono per certi aspetti equivalenti, cioè per tutti vale un teorema di correttezza e completezza e quindi l'insieme di teoremi che si può derivare con ciascuno di essi è lo stesso.

4.1 Sistemi assiomatici

L'approccio al problema della deduzione seguito nei sistemi assiomatici consiste nell'individuare un insieme di proposizioni da cui partire, come dei *principi* che ci permettano di ottenere altre proposizioni e di limitare al minimo il numero di regole di inferenza. Le proposizioni da cui partire, che quindi vengono assunte come

primitive sono gli *assiomi (logici)*.

Sono stati proposti vari sistemi assiomatici. Presentiamo qui di seguito quello introdotto da Hilbert e noto come *sistema hilbertiano* per la logica proposizionale.

Definizione 4.1 *Siano A, B e C tre simboli proposizionali. Il calcolo proposizionale è definito dai seguenti schemi di assioma e dalle seguenti regole di inferenza:*

1. *Schemi di assioma:*

   (a) $(A \to (B \to A))$;
   (b) $((A \to (B \to C)) \to ((A \to B) \to (A \to C)))$;
   (c) $((B \to \neg A) \to ((B \to A) \to \neg B))$.

2. *Regole di Inferenza:*

   - Modus Ponens (MP):
   $$\frac{A, \quad A \to B}{B}$$
   - Sostituzione Uniforme (SU):
   $$\frac{F[p]}{F[X/p]}$$

   dove p è un simbolo proposizionale che può occorrere in F, X è una proposizione e p non occorre in X, $F[X/p]$ è detta una istanza *di $F[p]$.*

Gli assiomi del sistema hilbertiano coinvolgono solo i connettivi $\to$ e $\neg$; vedremo nel seguito come estenderli agli altri connettivi. In effetti, è in virtù della sostituzione uniforme che possiamo parlare di schemi di assioma; in altri termini l'insieme AXIOMS degli assiomi del calcolo proposizionale è l'insieme di tutte le formule che si ottengono sostituendo uniformemente proposizioni al posto dei simboli proposizionali negli schemi *a, b* e *c*. È dunque chiaro che l'insieme degli assiomi AXIOMS è un insieme infinito, poiché l'insieme delle istanze di *a, b* e *c* è infinito.

Seguendo le definizioni 2.13 e 2.14, diciamo che una sequenza $A_1, \ldots, A_n$ è una *dimostrazione* di A se $A_n = A$ e se per ogni i, $1 \leq i \leq n$, A_i è un'istanza di uno schema di assioma (cioè è ottenuto da *a, b* e *c* per mezzo di SU), oppure A_i è ottenuto da A_j e A_k, $j < k < i$ per MP, dove $A_k = A_j \to A_i$. In tal caso A è un *teorema* del calcolo proposizionale e lo indichiamo:

$$\vdash_H A$$

che si legge "si dimostra A" e dove il pedice H sta a indicare che la dimostrazione è stata fatta con l'apparato deduttivo hilbertiano. Diciamo che una proposizione A ha una *prova da un insieme di proposizioni* Γ se esiste una derivazione $A_1, \ldots, A_n = A$ tale che, per ogni i, $1 \leq i \leq n$, A_i è un'istanza di uno schema di assioma (cioè è

ottenuto da a, b e c per mezzo di SU), oppure A_i è ottenuto da A_j e A_k, $j < k < i$ per MP, dove $A_k = A_j \to A_i$, o A_i è in Γ. Se A ha una dimostrazione da Γ scriviamo:

$$\Gamma \vdash_H A$$

che si legge "da Γ si deriva A". Gli elementi di Γ sono chiamati *le premesse* o *dipendenze* di A. Dalla definizione di dimostrazione, è evidente che

$$\vdash_H A \text{ implica } \Gamma \vdash_H A \text{ per qualunque } \Gamma$$

quindi, in particolare,

$$\Gamma \vdash_H A \text{ per ogni } A \in \text{AXIOMS}.$$

Nel seguito se non c'è possibilità di confusione omettiamo il pedice H; inoltre, se A ha una dimostrazione da $\Gamma \cup \{B\}$ scriviamo $\Gamma, B \vdash A$. Scriviamo $\Delta \vdash \Gamma$ per indicare che $\Delta \vdash A$ per ogni formula $A \in \Gamma$.

Illustriamo, ora, le proprietà della relazione di derivazione $\vdash$. Diciamo che $\vdash$ è una relazione, in quanto $\vdash \subseteq (\wp(\mathcal{L}) \times \mathcal{L})$.

Teorema 4.2 *Le seguenti sono proprietà della relazione di derivazione $\vdash$ nel calcolo proposizionale:*

1. *Inclusione: Se $A \in \Gamma$ allora $\Gamma \vdash A$;*
2. *Monotònia: Se $\Gamma \vdash A$ e $\Gamma \subseteq \Delta$ allora $\Delta \vdash A$;*
3. *Compattezza: $\Gamma \vdash A$ sse esiste un Γ_0 finito, tale che $\Gamma_0 \subseteq \Gamma$ e $\Gamma_0 \vdash A$;*
4. *Taglio: Se $\Delta \vdash A$ e per ogni $B \in \Delta$, $\Gamma \vdash B$ allora $\Gamma \vdash A$.*

Dimostrazione

1. *Inclusione.* Sia $A \in \Gamma$, dalla definizione di dimostrazione di A da Γ segue che $\Gamma \vdash A$.
2. *Monotònia.* Sia $\Gamma \vdash A$, dunque esiste una dimostrazione $A_1, \ldots, A_n = A$, da Γ per A. Procediamo per induzione completa.
   (Passo base). $A_1 \in \Gamma$ oppure A_1 è un assioma. Se $A_1 \in \Gamma \subseteq \Delta$ allora $A_1 \in \Delta$; se A_1 è una istanza di assioma allora $\Delta \vdash A_1$.
   (Passo induttivo). Supponiamo vero per ogni $m < n$ che $\Delta \vdash A_m$. Se $A_n \in \Gamma$ o A_n è un'istanza di assioma allora – come sopra – $\Delta \vdash A_n$. Se A_n è stato ottenuto mediante MP, è stato ottenuto da $A_j, A_k = A_j \to A_n$, $k, j < n$, ma per ipotesi induttiva $\Delta \vdash A_j$ e $\Delta \vdash A_j \to A_n$, pertanto per MP, $\Delta \vdash A_n$.
3. *Compattezza.* Se $\Gamma \vdash A$ allora esiste una dimostrazione $A_1, \ldots, A_n = A$ da Γ. Se nessun A_i è in Γ, allora $\vdash A$ e quindi la tesi. Altrimenti, consideriamo l'insieme degli A_i che occorrono nella dimostrazione di A tali che $A_i \in \Gamma$. Tale insieme è finito e lo chiamiamo Γ_0. Siccome per ogni altro A_j che occorre nella

dimostrazione di A abbiamo che o è una istanza di uno schema di assioma o è ottenuto da due precedenti proposizioni per MP, ne segue che $\Gamma_0 \vdash A$. Viceversa, supponiamo che $\Gamma_0 \vdash A$; per monotònia, poiché $\Gamma_0 \subseteq \Gamma$ abbiamo $\Gamma \vdash A$. (Si può notare che una direzione della compattezza è per l'appunto la monotònia).

4. *Taglio.* Supponiamo $\Delta \vdash A$ e per ogni $B \in \Delta$, $\Gamma \vdash B$. Consideriamo una derivazione $A_1, \ldots, A_n = A$ di A da Δ. Se Δ è vuoto la tesi è dimostrata. Altrimenti, per ogni $A_i \in \Delta$, avremo per ipotesi una dimostrazione $A_{k_1}, \ldots, A_{k_i} = A_i$ da Γ. Possiamo quindi sostituire in $A_1, \ldots, A_n = A$ ciascun $A_i \in \Delta$ con la relativa dimostrazione a partire da Γ, ottenendo la sequenza $A_1, \ldots, A_{k_1}, \ldots, A_{k_i} = A_i, \ldots, A_{l_1}, \ldots, A_{l_j} = A_j, \ldots A_n = A$. Questa sequenza è una dimostrazione di A da Γ, quindi $\Gamma \vdash A$.

□

Esempio 35 *La monotònia, informalmente, dice che al crescere delle conoscenze non decresce il numero di cose che possiamo derivare. Consideriamo, infatti, le seguenti affermazioni:*

tira-vento $\vdash$ *ho-freddo;*

piove e tira-vento $\vdash$ *ho-freddo.*

Se basta il vento a farmi sentire freddo, è evidente che se piove e tira vento a maggior ragione avrò freddo.

Tuttavia, la monotònia non è sempre intuitiva dal punto di vista del nostro senso comune. Consideriamo, infatti, le seguenti affermazioni:

zucchero-nel-caffè $\vdash$ *caffè-dolce;*

zucchero-nel-caffè e sale-nel-caffè $\vdash$ *caffè-dolce.*

È chiaro che il secondo enunciato è falso dal punto di vista del senso comune, tuttavia dal punto di vista della logica è vero.

Come accennato in precedenza, altri connettivi possono essere introdotti mediante definizioni, ad esempio:

1. $A \wedge B =_{Def} \neg(A \rightarrow \neg B)$;
2. $A \vee B =_{Def} \neg A \rightarrow B$;
3. $A \leftrightarrow B =_{Def} (A \rightarrow B) \wedge (B \rightarrow A)$.

Osserviamo che, in questo contesto, abbiamo dato delle definizioni per i connettivi perché non possiamo utilizzare la nozione di completezza di insiemi di connettivi introdotta nel paragrafo 3.4, che si basa sulla nozione di valutazione booleana. Qui, infatti, stiamo solo considerando un "calcolo", che è sintattico, dunque non siamo autorizzati a utilizzare nozioni semantiche, quali l'equivalenza tautologica[1].

[1]Fintanto che non dimostreremo un teorema che ci dice che "possiamo utilizzare le tautologie come teoremi", ovvero il teorema di completezza.

Teorema 4.3 [TEOREMA DI DEDUZIONE] *Sia Γ un insieme di formule proposizionali e siano A e B due formule proposizionali:*

$$\Gamma, B \vdash A \text{ sse } \Gamma \vdash B \to A.$$

Dimostrazione ($\Rightarrow$) Sia $A_1, \ldots, A_n = A$ una dimostrazione di A da $\Gamma \cup \{B\}$. Procediamo per induzione completa.
(*Passo Base*) A_1 è un assioma, oppure sta in Γ, oppure è B stesso. Per lo schema *(a)* abbiamo $A_1 \to (B \to A_1)$, dunque se A_1 è in Γ o A_1 è un assioma, abbiamo, per MP che $\Gamma \vdash B \to A_1$. Se $A_1 = B$, siccome $B \to B$ è un teorema (si veda l'esercizio 91), abbiamo $\Gamma \vdash B \to B$.
(*Passo Induttivo*) Supponiamo che $\Gamma \vdash B \to A_i$, $i < n$. Consideriamo A_n. Come sopra, se $A_n \in \Gamma$ o A_n è un assioma o A_n è B abbiamo $\Gamma \vdash B \to A_n$. Supponiamo allora che A_n sia stato ottenuto per MP. Dunque è stato ottenuto da due formule A_j e A_k, $j, k < n$. Per ipotesi induttiva $\Gamma \vdash B \to A_j$ e $\Gamma \vdash B \to A_k$. Ma A_k deve essere della forma $A_j \to A_n$, dunque abbiamo $\Gamma \vdash B \to A_j$ e $\Gamma \vdash B \to (A_j \to A_n)$. Per lo schema di assioma *(b)* abbiamo che $((B \to (A_j \to A_n)) \to ((B \to A_j) \to (B \to A_n)))$. Da questa formula e da $B \to A_j$ e $B \to (A_j \to A_n)$, applicando due volte MP, si avrà $\Gamma \vdash B \to A_n$.
($\Leftarrow$) Questa parte è lasciata come esercizio al lettore (si veda l'esercizio 94). □

Esempio 36 *Vogliamo mostrare che, nel sistema proposto si deriva:*

$$\vdash \neg A \to (A \to B)$$

1.	$\neg A$	*(ipotesi)*
2.	A	*(ipotesi)*
3.	$A \to (\neg B \to A)$	*(schema (a))*
4.	$\neg A \to (\neg B \to \neg A)$	*(schema (a))*
5.	$\neg B \to A$	*(da 2 e 3 per MP)*
6.	$\neg B \to \neg A$	*(da 1 e 4 per MP)*
7.	$(\neg B \to \neg A) \to ((\neg B \to A) \to \neg\neg B)$	*(schema (c))*
8.	$(\neg B \to A) \to \neg\neg B$	*(da 6 e 7 per MP)*
9.	$\neg\neg B$	*(da 5 e 8 per MP)*
10.	B	*(da 9 per esercizio 92)*

Questo, con due applicazioni del teorema di deduzione, conclude la dimostrazione.

Il teorema di deduzione ci dice che una formula può passare da sinistra a destra della relazione $\vdash$ introducendo il connettivo $\to$, oppure da destra a sinistra eliminandolo. Questo risultato è importante per due ragioni. In primo luogo perché ci fornisce la possibilità di esprimere una relazione metalinguistica (quale $\vdash$) mediante una formula del linguaggio.
In secondo luogo esso ci fornisce la possibilità di trattare con teoremi invece che con generiche derivazioni da insiemi di formule. Infatti, in virtù della proprietà di

compattezza di $\vdash$, abbiamo che se $\Gamma \vdash C$ allora esiste un insieme finito $\Gamma_0 \subseteq \Gamma$ tale che $\Gamma_0 \vdash C$. Siccome Γ_0 è finito sarà del tipo $\{A_1, \ldots, A_n\}$ o equivalentemente $\{A_1 \wedge \ldots \wedge A_n\}$, quindi: $\Gamma \vdash C$ implica $\Gamma_0 \vdash C$ che equivale a $(A_1 \wedge \ldots \wedge A_n) \vdash C$. Per il teorema di deduzione, otteniamo $\vdash (A_1 \wedge \ldots \wedge A_n) \rightarrow C$. Dall'esercizio 93 parte 3 abbiamo $\vdash A_1 \rightarrow (\ldots \rightarrow (A_n \rightarrow C)\ldots)$.

Introduciamo ora la nozione di *consistenza*.

Definizione 4.4 *Diciamo che un insieme di formule proposizionali* Γ è consistente *se è verificata una delle seguenti:*

1. *Non esiste una proposizione* A *tale che* $\Gamma \vdash A$ *e* $\Gamma \vdash \neg A$*;*
2. *Esiste una proposizione* A*, tale che* $\Gamma \nvdash A$*.*

Si può dimostrare (si veda l'esercizio 96) che le due formulazioni sono equivalenti. Si può inoltre correlare la nozione sintattica di consistenza con quella semantica di soddisfacibilità (introdotta in 3.8), dimostrando che un insieme di formule Γ è consistente sse è soddisfacibile (si veda l'esercizio 97).

Definizione 4.5 *Un insieme di formule* Γ *si dice* consistente e massimale *o* completo *se per ogni formula* $A \in \mathcal{L}$ *si verifica una e una sola delle seguenti relazioni:*

$$\Gamma \vdash A \textit{ oppure } \Gamma \vdash \neg A$$

Notare che la nozione di insieme consistente e massimale introdotta nella definizione 4.5 è analoga a quella di insieme massimale soddisfacibile introdotta nella definizione 3.22.

4.1.1 Esercizi

Nota: Negli esercizi di questo gruppo (91 – 96) il simbolo $\vdash$ sta per $\vdash_H$.

Esercizio 91 *Dimostrare che* $\vdash A \rightarrow A$*.*

Esercizio 92 *Dimostrare che* $\vdash \neg\neg A \rightarrow A$*.*

Esercizio 93 *Dimostrare che:*

1. $A \rightarrow B \vdash \neg B \rightarrow \neg A$*;*
2. $A \rightarrow B \rightarrow C \vdash B \rightarrow A \rightarrow C$*;*
3. $(A \wedge B) \rightarrow C \vdash A \rightarrow B \rightarrow C$*.*

Esercizio 94 *Dimostrare, utilizzando il teorema di deduzione:*

1. $A \rightarrow B, B \rightarrow C \vdash A \rightarrow C$*;*

2. $A \to B \to C, B \vdash A \to C$.

Esercizio 95 *Dimostrare la parte* ($\Leftarrow$) *del teorema di deduzione: dato un insieme di formule proposizionali* Γ *e due proposizioni* A *e* B*:*

$$\Gamma \vdash B \to A \textit{ implica } \Gamma, B \vdash A.$$

Esercizio 96 *Dimostrare che, nella definizione 4.4, 1 implica 2 e 2 implica 1.*

Esercizio 97 *Utilizzando una opportuna definizione di consistenza mostrare che un insieme di formule* Γ *è consistente se e solo se è soddisfacibile.*

4.2 Completezza e correttezza del sistema hilbertiano

Abbiamo introdotto, nel paragrafo precedente, le nozioni di "dimostrazione", "teorema", "assioma", "derivazione", eccetera, e abbiamo mostrato delle utili proprietà della relazione $\vdash_H$.

Illustriamo ora la relazione che lega i teoremi e le tautologie. La domanda che ci poniamo è se tutto ciò che deriviamo mediante $\vdash_H$ senza premesse è vero in tutti i modelli della logica.

In secondo luogo vogliamo sapere in che relazione stanno le formule proposizionali derivabili da un insieme di proposizioni Γ con le formule vere nei modelli di Γ. Supponiamo che Γ sia un insieme di proposizioni che descrivono una determinata realtà; quando deriviamo un enunciato A da Γ è importante sapere se A è vero nelle realtà che sono modelli di Γ. Ovvero ci interessa sapere se c'è una corrispondenza effettiva tra ciò che consegue dalle premesse e ciò che è vero nelle strutture che interpretano tali premesse.

Per mostrare questa corrispondenza, dobbiamo mostrare che:

(i) L'insieme delle tautologie coincide con l'insieme dei teoremi:

$$\vdash A \text{ sse } \models A.$$

(ii) Ciò che deriva da un insieme Γ è vero in tutti i modelli di Γ:

$$\Gamma \vdash A \text{ sse } \Gamma \models A.$$

La proprietà al punto i è detta *correttezza e completezza debole*, mentre quella al punto ii è detta *correttezza e completezza forte*. In effetti, per le logiche in cui – come nel nostro caso – vale un teorema di compattezza e di deduzione, i due enunciati sono equivalenti: $\Gamma \vdash B$ può essere scritto come $\vdash (A_1 \wedge \ldots \wedge A_n) \to B$,

dove $(A_1 \wedge \ldots \wedge A_n)$ è la congiunzione delle formule di Γ. La cosa si può facilmente verificare come mostrato a pagina 88.

Pertanto ci limitiamo a dimostrare:

$$\vdash_H A \to B \text{ sse } \models A \to B$$

Dire che l'insieme dei teoremi è contenuto nell'insieme delle tautologie assicura che l'apparato deduttivo è corretto: non può infatti dimostrare cose false. Dire che l'insieme delle tautologie è contenuto nell'insieme dei teoremi assicura che l'apparato deduttivo è completo, cioè che tutte le formule valide sono dimostrabili.

L'insieme dei teoremi del sistema deduttivo hilbertiano può essere definito induttivamente: esso è l'insieme generato dagli schemi di assioma e chiuso rispetto alle regole di MP e SU. Pertanto la correttezza si può mostrare induttivamente.

Più difficile da dimostrare è invece la completezza, perché l'insieme delle tautologie non è definito induttivamente.

Teorema 4.6 [CORRETTEZZA DEL SISTEMA HILBERTIANO]

$$\vdash A \to B \textit{ implica } \models A \to B.$$

Dimostrazione È sufficiente dimostrare che:

i. gli schemi *a, b* e *c* sono tautologie;

ii. MP e SU sono chiusi rispetto alla conseguenza tautologica. Ovvero se A e $A \to B$ sono tautologie anche B è una tautologia, e se $F[p]$ è una tautologia, anche $F[X/p]$ lo è.

La verifica che *a, b* e *c* sono tautologie è lasciata al lettore (si veda l'esercizio 99).

Verifichiamo che MP è chiuso rispetto alla conseguenza tautologica. Supponiamo per assurdo che $\models A$ e $\models A \to B$, ma $\not\models B$. Se $\not\models B$, allora esiste un modello $\mathcal{M}$ che non soddisfa B; ma A è una tautologia, perciò è verificata in ogni modello e quindi in $\mathcal{M}$. Pertanto $\mathcal{M} \models A$ e $\mathcal{M} \not\models B$ e per definizione di implicazione tautologica $\not\models A \to B$; contraddizione.

Per quanto riguarda la SU, sia $F[p]$ una tautologia, allora $\models F[p]$ per qualunque assegnazione booleana alle lettere proposizionali di $F[p]$, in particolare a p; dunque per qualunque proposizione X possiamo porre $I_\nu(p) = I_\nu(X)$. Per il teorema del rimpiazzamento, $I_\nu(F[p/p]) = I_\nu(F[X/p])$, da cui segue che $F[X/p]$ è una tautologia. □

Prima di dimostrare il teorema di completezza diamo alcune utili definizioni.

Definizione 4.7 *Sia* Γ *un insieme di formule proposizionali. Sia* $Cn(\Gamma)$ *l'insieme delle formule che hanno una derivazione da* Γ*, cioè* $Cn(\Gamma) = \{A | \Gamma \vdash A\}$.
$Cn(\Gamma)$ *è detto la* chiusura deduttiva *di* Γ.

Definizione 4.8 *Un insieme di formule proposizionali* Γ *è detto* deduttivamente chiuso *se esso coincide con la sua chiusura deduttiva, cioè* $\Gamma = Cn(\Gamma)$.

Teorema 4.9 [COMPLETEZZA DEL SISTEMA HILBERTIANO]

$$\models A \to B \ \textit{implica} \ \vdash_H A \to B.$$

Dimostrazione Scriviamo la tesi nella sua forma contrappositiva, ovvero:

$$\nvdash_H A \to B \ \textit{implica} \ \not\models A \to B.$$

Mostriamo che se $A \to B$ non è un teorema allora $A \to B$ non è una formula valida, cioè B non segue tautologicamente da A. A tal fine basta costruire, analogamente a quanto fatto per il teorema 3.24, un insieme Δ (deduttivamente chiuso e completo, cioè consistente e massimale, si veda la definizione 4.5) dal quale $A \to B$ non è derivabile. Mostreremo quindi che Δ ha un modello che soddisfa A e non soddisfa B; dunque $\not\models A \to B$.

Per costruire l'insieme Δ procediamo nel seguente modo. Enumeriamo tutte le formule del linguaggio:

$$A_0, A_1, A_2, A_3 \ldots$$

Definiamo, ora, una sequenza di insiemi di formule, nel seguente modo:

$$\begin{array}{lll} \Delta_0 & = \{A\} & \\ \Delta_{i+1} & = \begin{cases} \Delta_i & \text{se } \Delta_i, A_i \vdash B \\ \Delta_i \cup \{A_i\} & \text{se } \Delta_i, A_i \nvdash B \end{cases} \end{array}$$

per tutti gli $i > 0$. Osserviamo innanzi tutto che $\Delta_i \nvdash B$, per ogni i. Infatti, per induzione abbiamo che:
(*Passo Base*) $\Delta_0 \nvdash B$ perché $\Delta_0 = \{A\}$ e per ipotesi $A \to B$ non è un teorema.
(*Passo Induttivo*) Per costruzione dei Δ_i.
Sia, ora,

$$\Delta = \bigcup \Delta_i$$

Osserviamo che:

1. Per costruzione, $A \in \Delta$.
2. $\Delta \nvdash B$. Per dimostrare ciò, osserviamo innanzitutto che B non è un teorema, infatti, se lo fosse avremmo $\vdash B$ e per monotònia, $A \vdash B$; quindi per il teorema di deduzione, $\vdash A \to B$, contraddicendo l'ipotesi. Supponiamo dunque che B non sia un teorema e che $\Delta \vdash B$; esiste quindi una derivazione $B_1, \ldots B_n = B$ in Δ con qualche $B_j \in \Delta$. L'insieme di tali B_j è finito, quindi esiste un Δ_k che li include tutti. Dunque $\Delta_k \vdash B$, contraddizione. Siccome c'è una proposizione che non deriva da Δ, Δ è consistente (si veda la definizione 4.4, parte 2).
3. $B \notin \Delta$, per la proprietà di inclusione di $\vdash$.
4. Δ è deduttivamente chiuso, cioè per ogni formula F:

$$F \in \Delta \text{ sse } \Delta \vdash F.$$

Se $F \in \Delta$ allora $\Delta \vdash F$ è la proprietà di inclusione.

Ci rimane quindi da dimostrare che se $\Delta \vdash F$ allora $F \in \Delta$.

Supponiamo $\Delta \vdash F$, allora esiste una dimostrazione in Δ per F. Sia essa $F_1, \ldots, F_n = F$. Poiché l'enumerazione $A_0, A_1, A_2, \ldots$ comprende tutte le formule, F sarà un A_i. Se $F = A_i \notin \Delta$, allora $F = A_i \notin \Delta_{i+1}$ e quindi, per costruzione, $\Delta_i, F \vdash B$. Esistono, pertanto, delle formule $B_1, \ldots, B_m$ in Δ_i, tali che, $B_1, \ldots, F_1, \ldots, F_n \ldots, B_m$ costituiscono una dimostrazione in Δ per B, contraddicendo il punto 2.

5. Dimostriamo ora che l'insieme Δ è massimale, ovvero per ogni formula F:

$$F \in \Delta \text{ oppure } \neg F \in \Delta.$$

Sia $F \notin \Delta$ e supponiamo per assurdo $\neg F \notin \Delta$. Allora per qualche i e j: $\Delta_i, F \vdash B$ e $\Delta_j, \neg F \vdash B$. Per il teorema di deduzione e per il fatto che Δ_i e Δ_j sono contenuti in Δ abbiamo $\Delta \vdash F \to B$ e $\Delta \vdash \neg F \to B$ e, quindi (si veda l'esercizio 93) $\Delta \vdash \neg B \to \neg F$ e $\Delta \vdash \neg B \to F$.

Istanziando lo schema c, mediante SU, abbiamo: $(\neg B \to \neg F) \to ((\neg B \to F) \to B)$. Applicando MP due volte otteniamo $\Delta \vdash B$; contraddizione con il punto 2.

6. Il punto 5 ci dice anche che Δ è chiuso rispetto alla negazione. Dimostriamo ora che l'insieme Δ è chiuso rispetto all'implicazione, cioè

$$F \to G \in \Delta \text{ sse } F \notin \Delta \text{ oppure } G \in \Delta.$$

($\Rightarrow$) Dimostriamo che $F \to G \in \Delta$ implica $F \notin \Delta$ oppure $G \in \Delta$ per assurdo. Supponiamo che $F \to G \in \Delta$, $F \in \Delta$ e $G \notin \Delta$. Per inclusione abbiamo che $\Delta \vdash F \to G$ e $\Delta \vdash F$, pertanto per MP, $\Delta \vdash G$ e, per la chiusura deduttiva di Δ, $G \in \Delta$; contraddizione.
($\Leftarrow$) Dimostriamo che $F \notin \Delta$ oppure $G \in \Delta$ implica $F \to G \in \Delta$.
Dobbiamo considerare i due casi: $F \notin \Delta$ oppure $G \in \Delta$. Supponiamo, per primo, che $G \in \Delta$. Da questo, per inclusione abbiamo $\Delta \vdash G$. Istanziando lo schema di assioma a, mediante SU, abbiamo, $G \to (F \to G)$. Dunque per MP, $\Delta \vdash F \to G$ ne segue, per la chiusura deduttiva di Δ, che $F \to G \in \Delta$.
Supponiamo, ora, che $F \notin \Delta$. Per il punto 5, $\neg F \in \Delta$. Ricordiamo dall'esempio 36 che $(\neg F \to (F \to G))$ è un teorema, cioè : $\vdash (\neg F \to (F \to G))$. Per MP abbiamo $F \to G \in \Delta$.

Essendo Δ consistente, esso ha almeno un modello. Costruiamo quindi un modello $\mathcal{M}$ per Δ. Partiamo da una assegnazione ai simboli proposizionali:

$$P \in \mathcal{M} \text{ sse } P \in \Delta.$$

Si può verificare, per induzione strutturale sulle formule, che per ogni F:

$$\mathcal{M} \models F \text{ sse } F \in \Delta.$$

Questa verifica è lasciata al lettore (si veda l'esercizio 100).
Da ciò deduciamo che, poiché $B \notin \Delta$, allora $\mathcal{M} \not\models B$ e, d'altro canto, poiché $A \in \Delta$, $\mathcal{M} \models A$; da cui segue che $\mathcal{M} \not\models A \to B$ e pertanto $A \to B$ non è una tautologia, ovvero $\not\models A \to B$. Questo conclude la dimostrazione. □

Osserviamo che, per mostrare la completezza, abbiamo utilizzato (esplicitamente o per aver utilizzato il teorema di deduzione) tutti gli schemi di assioma *a, b* e *c* del sistema hilbertiano, insieme a MP e a SU. Se un assioma non fosse stato utilizzato in maniera essenziale nella dimostrazione, questo avrebbe significato che esso non è di fatto necessario a mostrare che ciò che si deriva dal sistema deduttivo è verificato nei modelli della logica. Se un assioma non è necessario può, eventualmente, essere derivato, quindi di fatto non è un assioma, perché consegue dagli altri assiomi.

L'apparato deduttivo hilbertiano, è importante perché fornisce una nozione rigorosa di derivazione e, soprattutto, fornisce una costruzione induttiva dell'insieme dei teoremi. In questo senso è elegante e chiaro.

Tuttavia in intelligenza artificiale a esso sono preferiti altri apparati deduttivi, per un motivo che spieghiamo brevemente.
Il sistema hilbertiano richiede, per essere automatizzato, una notevole euristica (si veda l'esempio 36). Infatti, per costruire una derivazione è necessario sapere quali ipotesi, quali assiomi e quali teoremi utilizzare. Inoltre, a ogni passo della dimostrazione qualunque teorema può essere introdotto e, quindi, la lunghezza e la complessità di una dimostrazione dipendono dall'efficacia dell'euristica. Il problema è che non c'è alcuna misura per garantirsi che i passi successivi di una dimostrazione diventino via via più semplici. Per questo motivo sono stati introdotti apparati deduttivi che si basano su un principio fondamentale: ogni passo di derivazione deve semplificare le proposizioni che stiamo verificando. Questi apparati deduttivi saranno oggetto dei prossimi paragrafi.

4.2.1 Esercizi

Esercizio 98 *Analogamente a quanto fatto nel lemma 3.27 per le conseguenze tautologiche, mostrare che se* Γ *è effettivamente enumerabile allora l'insieme delle formule dimostrabili da* Γ*, ovvero l'insieme* $\{A|\Gamma \vdash A\}$*, è effettivamente enumerabile.*

Esercizio 99 *Verificare che gli schemi di assioma a,b e c sono tautologie.*

Esercizio 100 *Completare la dimostrazione della costruzione del modello nel teorema di completezza: mostrare per induzione strutturale sulle formule, che per ogni* F*:*

$$\mathcal{M} \models F \text{ sse } F \in \Delta.$$

Esercizio 101 *In riferimento al teorema di completezza, è necessario mostrare che* Δ *è massimale. Dove si sfrutta tale argomento?*

4.3 Deduzione Naturale

La deduzione naturale, introdotta da Gentzen, studiata in particolare da Prawitz [37], è un apparato inferenziale per il calcolo proposizionale basato sull'idea intuitiva che con le regole di inferenza si possa caratterizzare il "comportamento" dei singoli connettivi logici. Consideriamo un calcolo proposizionale in cui si abbiano la costante $\bot$ e i connettivi $\{\neg, \wedge, \vee, \rightarrow\}$ e gli altri siano introdotti per definizione.

Analizziamo dapprima la congiunzione. Se si ha una dimostrazione sia per A che per B, allora si ha una dimostrazione per $A \wedge B$. Questo, scritto sotto forma di regola di inferenza diventa:

$$(\wedge i) \quad \frac{A \quad B}{A \wedge B}$$

Questa regola si chiama di *introduzione dell'and*: $(\wedge i)$.

Analogamente, se si ha una dimostrazione per $A \wedge B$, si ha una dimostrazione per A e anche per B. Si hanno quindi due regole di *eliminazione dell'and*: $(\wedge e_1)$ e $(\wedge e_2)$.

$$(\wedge e_1) \quad \frac{A \wedge B}{A}$$

$$(\wedge e_2) \quad \frac{A \wedge B}{B}$$

Un analogo ragionamento ci porta a considerare due regole per l'*introduzione dell'or*: $(\vee i_1)$ e $(\vee i_2)$.

$$(\vee i_1) \quad \frac{A}{A \vee B}$$

$$(\vee i_2) \quad \frac{B}{A \vee B}$$

Rimandiamo la trattazione di una regola per l'eliminazione dell'or, in quanto ci richiede qualche considerazione aggiuntiva.

Occupiamoci invece della *eliminazione dell'implicazione*.

$$(\rightarrow e) \quad \frac{A \;\; A \rightarrow B}{B}$$

Questa regola è il *Modus Ponens*.

Introduciamo ora la regola per l'*eliminazione della negazione* $(\neg e)$. Essa corrisponde all'intuizione che se abbiamo dimostrato sia A che $\neg A$, allora si ha una contraddizione.

$$(\neg e) \quad \frac{A \;\; \neg A}{\bot}$$

La regola per l'*eliminazione della contraddizione* $(\bot e)$ invece rispecchia l'intuizione che "ex falso quodlibitur", cioè dal falso si può concludere qualunque cosa.

$$(\bot e) \quad \frac{\bot}{A}$$

Le regole fin qui esposte vengono chiamate *elementari*. Nel seguito introdurremo invece le regole *condizionali*. Per spiegare il significato di questo nome facciamo ricorso a nozioni di tipo semantico. L'applicabilità di una regola elementare si basa solo sul fatto che le sue premesse siano vere. L'applicazione di una regola elementare implica che la verità della conclusione dipende solo dalla verità delle premesse, quindi le "dipendenze"– cioè l'insieme delle formule da cui dipende la verità – della conclusione sono l'unione delle dipendenze delle premesse.

Le regole che mostriamo qui di seguito invece si chiamano *condizionali*, in quanto si pretende non solo che le premesse siano vere, ma che (alcune di) esse siano state dimostrate a partire da certe ipotesi.

Prendiamo ad esempio la regola di *eliminazione dell'or*.

$$(\vee e) \quad \frac{A \vee B \quad \begin{matrix}[A] \\ C\end{matrix} \quad \begin{matrix}[B] \\ C\end{matrix}}{C}$$

Essa dice che: se abbiamo una prova per C a partire da un'ipotesi A, abbiamo una prova per C a partire da un'ipotesi B e inoltre abbiamo una prova per $A \vee B$, possiamo concludere C scartando le dipendenze A e B, cioè facendo dipendere la verità di C non da A e B presi singolarmente, ma da $A \vee B$. Le ipotesi che possono essere scartate vengono di solito scritte tra parentesi quadre. La verità di C verrà quindi a dipendere dalle ipotesi da cui eventualmente dipende la verità di $A \vee B$.

Una dimostrazione che si basi sull'uso della regola di eliminazione dell'or di fatto corrisponde a una dimostrazione per casi: se vogliamo fornire una prova che C segue da $A \vee B$, basta dimostrare che C si può derivare in ciascuno dei due casi, sia che A sia vero, sia che sia vero B, cioè C segue da A e C segue da B.

Analoga intuizione soggiace alla regola di *introduzione dell'implicazione*.

$$(\rightarrow i) \quad \frac{\begin{matrix}[A] \\ B\end{matrix}}{A \rightarrow B}$$

Questa regola corrisponde a una forma di teorema di deduzione: se abbiamo una prova di B a partire da A possiamo concludere $A \rightarrow B$, scartando la dipendenza da A. Quindi la nozione metalinguistica di deduzione $\vdash$ viene espressa nel linguaggio mediante $\rightarrow$.

Concludiamo la presentazione delle regole della deduzione naturale mostrando le regole di *introduzione della negazione* e della *Reductio ad Absurdum*.

$$(\neg i) \quad \frac{\begin{matrix}[A] \\ \bot\end{matrix}}{\neg A}$$

$$(RA) \quad \frac{\begin{matrix}[\neg A] \\ \bot\end{matrix}}{A}$$

La regola (RA) è il principio su cui si basa la dimostrazione per contraddizione: se si deve provare A si dimostra invece che $\neg A$ porta a una contraddizione[2].

Ogni regola di deduzione rappresenta un passo di inferenza. Qualora la conclusione di una regola coincida con la premessa di un'altra si possono applicare regole in sequenza dando luogo a una prova o derivazione. Una *prova* per una formula C che dipende da Γ è una sequenza finita di applicazioni di regole di inferenza in cui l'ultima formula è C e le formule intermedie sono conclusioni di regole di inferenza, oppure sono tra parentesi quadre, o sono in Γ. Scriviamo in questo caso:

$$\Gamma \vdash_{DN} C$$

e diciamo che C è derivabile nella deduzione naturale dalle premesse Γ. Se Γ è vuoto diciamo che la prova è una dimostrazione di C nella deduzione naturale, scriviamo

$$\vdash_{DN} C$$

e diciamo che C è un *teorema* della deduzione naturale. Le derivazioni fatte mediante deduzione naturale sono convenientemente rappresentabili in forma di albero la cui radice è etichettata con la formula dimostrata (è comodo in questo caso visualizzare l'albero con le foglie in alto e la radice in basso). Se nella dimostrazione di una formula C viene utilizzata una regola condizionale, le dipendenze da scartare – in accordo con quanto specificato nella regola – vengono marcate con parentesi quadre nel sottoalbero che porta a C. Se l'insieme delle formule associate ai nodi foglia dell'albero e non racchiuse tra parentesi quadre, cioè non scartate durante il processo di dimostrazione, è contenuto in Γ e la radice è etichettata C, l'albero rappresenta una dimostrazione di C a partire da Γ, cioè $\Gamma \vdash_{DN} C$. In particolare, se tutte le foglie sono state scartate durante le derivazione di C si ha che $\vdash_{DN} C$.

Diamo ora alcuni esempi di dimostrazioni in deduzione naturale.

Esempio 37 *Dimostriamo che:* $\vdash_{DN} A \to (B \to A)$.

$$\dfrac{\dfrac{[B]_1 \quad [A]_2}{(B \to A)}\ (\to i)}{A \to (B \to A)}\ (\to i)$$

Abbiamo evidenziato le regole utilizzate a ogni passo; le ipotesi sono state marcate con un pedice che indica l'ordine con cui sono state scartate.

Esempio 38 *Dimostriamo che:* $\vdash_{DN} A \to \neg\neg A$.

$$\dfrac{\dfrac{\dfrac{[A]_2 \quad [\neg A]_1}{\bot}\ (\neg e)}{\neg\neg A}\ (\neg i)}{A \to \neg\neg A}\ (\to i)$$

[2]Inserire (RA) tra le regole di inferenza significa costruire un sistema di logica classica. Omettendo invece (RA) (o regole a essa equivalenti) si costruisce un sistema di logica intuizionista.

Esempio 39 *Dimostriamo che* $\vdash_{DN} (A \to (B \to C)) \to (A \wedge B \to C)$

$$\dfrac{\dfrac{\dfrac{\dfrac{[A \wedge B]}{B} \quad \dfrac{\dfrac{[A \wedge B]}{A} \quad [A \to (B \to C)]}{B \to C}}{C}}{(A \wedge B \to C)}}{(A \to (B \to C)) \to (A \wedge B \to C)}$$

Con riferimento alla nozione di sistema formale data nel capitolo 2, si noti che la deduzione naturale è un sistema che si basa su un insieme di regole di inferenza, ma non presuppone un insieme di assiomi logici.

Si può dimostrare che le derivazioni in deduzione naturale possono essere espresse in una forma detta "normale", cioè considerando un qualunque cammino che va dalla radice verso una foglia, c'è una formula A in un nodo sul cammino tale che tutte le formule di qualunque cammino al di sopra di A (cioè verso le foglie) sono le premesse per regole di eliminazione, e tutte le formule al di sotto di A verso la radice, esclusa la radice stessa, sono le premesse per regole di introduzione. Quindi, nella parte alta dell'albero (sopra A), passiamo dalle ipotesi a sottoformule sempre più semplici, fino ad arrivare ad A, per poi proseguire nella composizione di formule sempre più complesse che contengono le precedenti come sottoformule. Quindi A è la formula "minima" dell'albero (si veda Prawitz [37]).

La deduzione naturale è un formalismo semplice e naturale per rappresentare dimostrazioni, ma piuttosto difficile da usare nella ricerca delle stesse. Questo perché non vale il *principio della sottoformula*, cioè – ad esempio nel caso di un connettivo binario $\circ$ – non è detto che per dimostrare $A \circ B$ si debba cercare una dimostrazione per A, una per B e si possa poi concludere con una regola di introduzione del connettivo $\circ$. Si potrebbero invece dover utilizzare regole quali (RA). Quindi la deduzione naturale mal si presta alla costruzione di dimostratori automatici di teoremi. Altri sistemi si prestano molto meglio a questo scopo, ad esempio i tableau e la risoluzione che vedremo nei prossimi paragrafi, o il calcolo dei sequenti (per il quale si rimanda per esempio a [3]). La deduzione naturale, invece, proprio per la forma intuitiva delle dimostrazioni si presta molto bene alla costruzione di dimostratori interattivi di teoremi.

Anche nel caso della deduzione naturale si può dimostrare che valgono le proprietà di inclusione, monotònia, compattezza e taglio.

Le proprietà di inclusione, compattezza e taglio sono ovvie e si possono dimostrare in modo del tutto analogo al caso del sistema hilbertiano.

La proprietà di monotònia nel caso della deduzione naturale sembra essere in conflitto con il fatto che le ipotesi "superflue" in una derivazione vengono racchiuse tra parentesi quadre e scartate, ma si noti che esse "possono" e non "debbono" essere scartate.

Anche nel caso della deduzione naturale si può dimostrare il teorema di deduzione.

Teorema 4.10 [TEOREMA DI DEDUZIONE] *Siano A e B due formule proposizionali:*

$$B \vdash A \text{ sse } \vdash B \to A.$$

Dimostrazione
($\Rightarrow$) Sia $A_1 \dots A_n = A$ una dimostrazione di A da B. Applicando la regola di introduzione dell'implicazione otteniamo la dimostrazione di $A_1 \dots A_n = A, A_{n+1} = B \to A$, infatti abbiamo scartato la dipendenza da B.
($\Leftarrow$) Sia $\vdash B \to A$; considerando B come ipotesi, con una applicazione della regola di eliminazione dell'implicazione, cioè del Modus Ponens, otteniamo una dimostrazione di A con premessa B. □

Ovviamente, il teorema di deduzione può essere esteso a

$$\Gamma, B \vdash A \text{ sse } \Gamma \vdash B \to A$$

per ogni insieme di formule Γ, per il teorema di compattezza.
La deduzione naturale, come il sistema hilbertiano può essere provata corretta e completa. Possiamo enunciare un teorema di correttezza e completezza debole che riguarda solo le tautologie:

Teorema 4.11 *A è una tautologia sse A ha una dimostrazione in deduzione naturale, cioè*

$$\vdash_{DN} A \text{ sse } \models A.$$

Possiamo anche dare anche una caratterizzazione *forte* di completezza. con Γ un insieme qualunque di formule.

Teorema 4.12 *Sia Γ un insieme di formule proposizionali:*

$$\Gamma \vdash_{DN} A \text{ sse } \Gamma \models A.$$

Come nel caso del sistema hilbertiano la correttezza e completezza forte si riducono alla correttezza e completezza debole.

Per la correttezza e completezza debole si può dare una dimostrazione diretta, oppure osservare che:

> *Correttezza*: Tutte le regole della deduzione naturale conservano la validità. Il lettore lo può verificare come esercizio (si veda l'esercizio 106).
>
> *Completezza*: Gli schemi di assioma a, b e c hanno una dimostrazione in deduzione naturale (si veda l'esercizio 106), MP corrisponde alla regola di eliminazione dell'implicazione e infine SU conserva la dimostrabilità in deduzione naturale (si veda l'esercizio 107). Quindi $\vdash_H$ implica $\vdash_{DN}$. Abbiamo quindi, per la completezza del sistema di Hilbert: $\models A$ implica $\vdash_H A$ implica $\vdash_{DN} A$.

4.3.1 Esercizi

Nota: Negli esercizi di questo gruppo (102 – 107) il simbolo $\vdash$ sta per $\vdash_{DN}$.

Esercizio 102 *Dimostrare che* $\vdash A \to A$ *è un teorema del calcolo proposizionale.*

Esercizio 103 *Dimostrare che:*

1. $A \to B \vdash \neg B \to \neg A$;
2. $A \to B \to C \vdash B \to A \to C$;
3. $(A \wedge B) \to C \vdash A \to B \to C$.

Esercizio 104 *Dimostrare, utilizzando il teorema di deduzione:*

1. $A \to B, B \to C \vdash A \to C$;
2. $A \to B \to C, B \vdash A \to C$.

Esercizio 105 *Dimostrare le proprietà di inclusione, monotònia, compattezza e taglio per la deduzione naturale.*

Esercizio 106 *Dimostrare che gli assiomi del sistema hilbertiano sono teoremi della deduzione naturale.*

Esercizio 107 *Dimostrare che se* $\Gamma \vdash F[p]$ *allora* $\Gamma \vdash F[X/p]$.

4.4 Tableau proposizionali

Presentiamo in questo paragrafo i tableau proposizionali. Essi sono stati introdotti da Hintikka e Beth alla fine degli anni '50 e poi ripresi da Smullyan in [47] e rielaborati successivamente da Fitting e Hähnle [19, 21].

Smullyan ha proposto uno schema per la decomposizione delle formule del calcolo proposizionale che le suddivide in due categorie: α e β. Secondo questo schema una formula del calcolo proposizionale, che non sia un letterale o un letterale negato, appartiene a una delle due seguenti categorie:

- α: formula *proposizionale congiuntiva*, la cui soddisfacibilità equivale alla soddisfacibilità di entrambe due sue componenti;
- β: formula *proposizionale disgiuntiva*, la cui soddisfacibilità equivale alla soddisfacibilità di almeno una tra due sue componenti.

In altre parole, lo schema raggruppa in due categorie tutte le formule del calcolo proposizionale del tipo $(A \circ B)$ e $\neg(A \circ B)$, con $\circ \in \{\wedge, \vee, \rightarrow\}$: quelle che agiscono *congiuntivamente* e sono chiamate *α-formule* e quelle che agiscono *disgiuntivamente* e sono chiamate *β-formule*. Ogni α-formula ha due componenti indicate con α_1 e α_2; ogni β-formula ha due componenti indicate con β_1 e β_2, rispettivamente, come da tabella qui di seguito riportata.

α	α_1	α_2	β	β_1	β_2
$A \wedge B$	A	B	$A \vee B$	A	B
$\neg(A \vee B)$	$\neg A$	$\neg B$	$\neg(A \wedge B)$	$\neg A$	$\neg B$
$\neg(A \rightarrow B)$	A	$\neg B$	$A \rightarrow B$	$\neg A$	B

Osserviamo che per ogni valutazione booleana $I_\mathcal{V}$ e per tutte le formule di tipo α e β si ha:

$$I_\mathcal{V}(\alpha) = I_\mathcal{V}(\alpha_1) \wedge I_\mathcal{V}(\alpha_2)$$
$$I_\mathcal{V}(\beta) = I_\mathcal{V}(\beta_1) \vee I_\mathcal{V}(\beta_2).$$

Inoltre, per ogni formula di tipo α e β si ha che $\alpha \leftrightarrow (\alpha_1 \wedge \alpha_2)$ e $\beta \leftrightarrow (\beta_1 \vee \beta_2)$ sono tautologie. Tali proprietà giustificano la nomenclatura per cui le formule α sono dette congiuntive e le β disgiuntive.

La decomposizione di Smullyan è usata per definire le corrispondenti regole per i tableau o *regole di espansione dei tableau*, dove la virgola va letta come un "and", la barra verticale come un "or":

$$\neg\text{-regole)} \quad \frac{\neg\neg A}{A} \qquad \frac{\neg\top}{\bot} \qquad \frac{\neg\bot}{\top}$$

$$\alpha\text{-regole)} \quad \frac{\alpha_1 \wedge \alpha_2}{\alpha_1, \alpha_2} \qquad \frac{\neg(\alpha_1 \vee \alpha_2)}{\neg\alpha_1, \neg\alpha_2} \qquad \frac{\neg(\alpha_1 \rightarrow \alpha_2)}{\alpha_1, \neg\alpha_2}$$

$$\beta\text{-regole)} \quad \frac{\beta_1 \vee \beta_2}{\beta_1 | \beta_2} \qquad \frac{\neg(\beta_1 \wedge \beta_2)}{\neg\beta_1 | \neg\beta_2} \qquad \frac{\beta_1 \rightarrow \beta_2}{\neg\beta_1 | \beta_2}$$

Tutte le regole possono essere espresse in forma compatta come segue:

$$1.\ \frac{\neg\neg A}{A} \quad 2.\ \frac{\neg\top}{\bot} \quad 3.\ \frac{\neg\bot}{\top} \quad 4.\ \frac{\alpha}{\alpha_1\ , \alpha_2} \quad 5.\ \frac{\beta}{\beta_1 | \beta_2} \tag{4.1}$$

Il connettivo $\leftrightarrow$ può essere trattato riscrivendo $A \leftrightarrow B$ come $(A \rightarrow B) \wedge (B \rightarrow A)$, ma si può anche introdurre una regola "composta" a esso specifica:

$\leftrightarrow$-regole)	$\dfrac{A \leftrightarrow B}{A, B \mid \neg A, \neg B}$	$\dfrac{\neg(A \leftrightarrow B)}{A, \neg B \mid \neg A, B}$

Nel seguito supponiamo che Γ sia un insieme finito di formule proposizionali.

Definizione 4.13 [TABLEAU] *Dato* Γ*, un* tableau *per* Γ *è un albero binario i cui nodi sono etichettati da formule di* Γ*.*

Definizione 4.14 *Dato* Γ*, l'albero binario costituito dal solo nodo radice etichettato dalla congiunzione delle formule di* Γ *è detto* tableau iniziale *per* Γ *e denotato* $\mathcal{T}_0$*.*

Definizione 4.15 [PASSO DI ESPANSIONE] *Dato* Γ*, se* $\mathcal{T}_1$ *è un tableau per* Γ *e* A *è il nodo foglia su un ramo* $\mathcal{B}$ *di* $\mathcal{T}_1$*, possiamo costruire un tableau* $\mathcal{T}_2$ *per* Γ *attraverso un* passo di espansione*:*

1. *Se* $\neg\neg B$*, oppure* $\neg\top$ *oppure* $\neg\bot$ *occorrono sul ramo* $\mathcal{B}$ *di* $\mathcal{T}_1$ *che porta alla foglia* A*, allora si aggiunge come figlio di* A *un ramo contenente* B*,* $\bot$ *oppure* $\top$*, rispettivamente (regole 1, 2 e 3 della tabella 4.1);*
2. *Se una formula di tipo* α *occorre sul ramo* $\mathcal{B}$ *di* $\mathcal{T}_1$ *che porta alla foglia* A*, allora si aggiunge come figlio di* A *un ramo contenente* α_1 *e* α_2 *in successione (regola 4 della tabella 4.1);*
3. *Se una formula di tipo* β *occorre sul ramo* $\mathcal{B}$ *di* $\mathcal{T}_1$ *che porta alla foglia* A*, allora si aggiungono come figli di* A *due rami contenenti rispettivamente* β_1 *e* β_2 *(regola 5 della tabella 4.1).*

Il tableau $\mathcal{T}_2$ *si dice ottenuto da* $\mathcal{T}_1$ con un passo di espansione*.*

Definizione 4.16 *Dati due tableau* $\mathcal{T}_1$ *e* $\mathcal{T}_2$ *per* Γ*,* $\mathcal{T}_2$ *è una* espansione coerente *di* $\mathcal{T}_1$ *se esiste un nodo* n *in* $\mathcal{T}_1$ *tale che* $\mathcal{T}_2$ *è stato ottenuto da* $\mathcal{T}_1$ *attraverso un numero finito di passi di espansione ognuno dei quali ha espanso la formula che etichetta* n *su tutte le foglie del sottoalbero che ha come radice* n*.*

Definizione 4.17 *Un tableau* $\mathcal{T}$ *per* Γ *si dice* ben costruito *se è stato ottenuto per espansioni coerenti dal tableau radice e nessun nodo è stato oggetto di più di una espansione coerente.*

Definizione 4.18 *Un tableau* $\mathcal{T}$ *per* Γ *è* completo *se è ben costruito e non può più essere oggetto di espansioni coerenti.*

Il ramo di un tableau può essere visto come la congiunzione delle formule che appaiono in esso e l'intero tableau come una disgiunzione di congiunzioni.

Definizione 4.19 *Un* ramo *di un tableau è* soddisfacibile *se la congiunzione delle formule che etichettano i suoi nodi è soddisfacibile.*

Definizione 4.20 *Un* tableau è soddisfacibile *se almeno uno dei suoi rami è soddisfacibile.*

Osserviamo che l'espansione preserva la soddisfacibilità, ovvero se un tableau è soddisfacibile il tableau ottenuto attraverso l'applicazione di una regola di espansione è ancora soddisfacibile.

Definizione 4.21 *Un* ramo *di un tableau si dice* chiuso *se, per qualche formula A, entrambe A e $\neg A$ etichettano nodi che occorrono sul ramo, oppure $\neg\top$ o $\bot$ occorrono sul ramo. Altrimenti il ramo è detto* aperto.

Definizione 4.22 *Un* tableau è chiuso *se tutti i suoi rami sono chiusi, altrimenti è* aperto.

Definizione 4.23 *Una* tableau-refutazione di una formula A *è un tableau chiuso la cui radice è etichettata da A.*

Definizione 4.24 *Una* tableau-refutazione di una formula A da Γ *è un tableau chiuso la cui radice è etichettata da $\Gamma \cup \{A\}$.*

Definizione 4.25 [TABLEAU-DIMOSTRAZIONE] *Una* tableau-dimostrazione *di una formula A è un tableau chiuso la cui radice è etichettata da $\neg A$. A è un* teorema del sistema di calcolo dei tableau *se A ha una tableau-dimostrazione. In questo caso scriviamo:*

$$\vdash_T A$$

Definizione 4.26 [TABLEAU-DEDUZIONE] *Una* tableau-deduzione *di una formula A da Γ, insieme finito di formule, è un tableau chiuso la cui radice è etichettata da $\Gamma \cup \{\neg A\}$. In questo caso scriviamo:*

$$\Gamma \vdash_T A$$

Il metodo dei tableau è facilmente meccanizzabile. Presentiamo qui di seguito un semplice algoritmo per costruire tableau-deduzioni.

Algoritmo 4.27 [COSTRUZIONE DI TABLEAU] Per costruire una tableau-deduzione per $\Gamma \vdash_T A$:

- Costruiamo il tableau $\mathcal{T}_0$ etichettato con $\Gamma \cup \{\neg A\}$.
- Finché ci sono nodi non marcati, scegliamo un nodo da espandere e
  - facciamo un passo di espansione coerente, cioè espandiamo il nodo rispetto a tutte le foglie a esso sottostanti nei rami ancora aperti;

- marchiamo il nodo espanso;
- verifichiamo se i rami interessati dall'espansione sono chiusi e li marchiamo tali.

Il procedimento si arresta quando tutti i rami sono stati chiusi oppure quando il tableau è completo. Se il tableau è chiuso, A è dedotto a partire da Γ. Altrimenti, se il tableau è completo e c'è ancora qualche ramo aperto, abbiamo provato che il tableau è consistente, quindi $\Gamma \cup \{\neg A\} \not\vdash_T$, cioè $\Gamma \not\vdash_T A$.

Osserviamo che questo algoritmo termina sempre (il lettore può dimostrarlo, si veda l'esercizio 108) e non necessariamente espande tutte le formule che compaiono nei nodi intermedi: l'inconsistenza infatti può essere individuata prima di arrivare a un tableau completo.

Osserviamo anche che l'algoritmo 4.27, a causa del nondeterminismo (nella scelta del nodo da espandere) è estremamente inefficiente. Una semplice euristica nella scelta dei nodi porta a utilizzare per prime le regole 1, 2 e 3 della tabella 4.1 e a dare la precedenza alle α-regole, in quanto questo permette di mantenere basso il numero dei nodi sulla frontiera dell'albero.

Esempio 40 *Mostriamo ora che dall'insieme di formule*

$$\Gamma = \{\neg(\neg P \vee \neg Q), P \wedge Q \rightarrow R\}$$

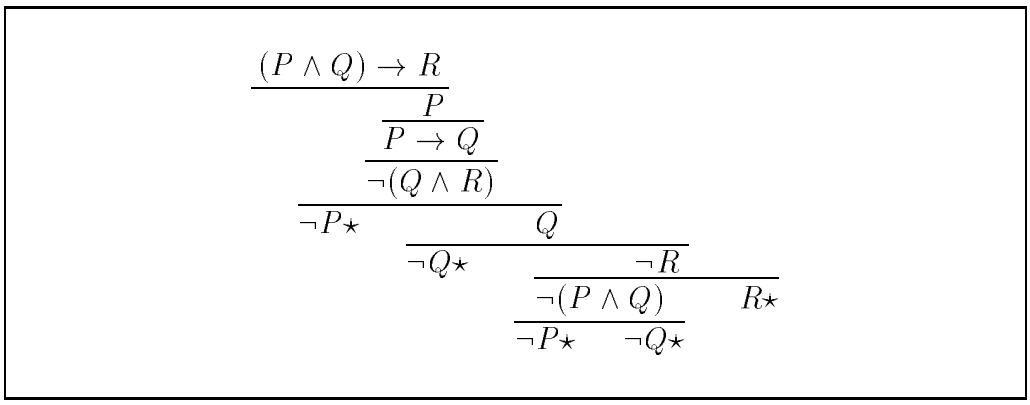
si deriva R. Abbiamo marcato con il simbolo $\star$ i rami chiusi. Facciamo osservare che l'uso di alcune regole (ad esempio la 1), è stato sottinteso.

$$\dfrac{\dfrac{\dfrac{\dfrac{\neg(\neg P \vee \neg Q), P \wedge Q \rightarrow R, \neg R}{\neg(\neg P \vee \neg Q)}}{P, Q}}{\dfrac{\neg(P \wedge Q) \qquad R\star}{\neg P\star \quad \neg Q\star}}}$$

Esempio 41 *Mostriamo ora che dall'insieme di formule*

$$\Gamma = \{P \wedge Q \rightarrow R, P, P \rightarrow Q\}$$

si deriva $Q \wedge R$.



Il sistema dei tableau è corretto e completo. Possiamo per esso enunciare un teorema di correttezza e completezza debole che riguarda solo le tautologie:

Teorema 4.28 *A è una tautologia sse A ha una tableau-dimostrazione, cioè*

$$\vdash_T A \text{ sse } \models A.$$

Possiamo anche dare anche una caratterizzazione *forte* di completezza. con Γ un insieme qualunque di formule, non necessariamente finito. Allo scopo diamo la seguente definizione.

Definizione 4.29 *Una formula A ha una* tableau deduzione *da Γ, insieme qualunque di formule, e scriviamo $\Gamma \vdash_T A$, sse esiste un Γ_0 finito, sottoinsieme di Γ, tale che $\Gamma_0 \vdash_T A$.*

Teorema 4.30 *Sia Γ un insieme di formule proposizionali:*

$$\Gamma \vdash_T A \text{ sse } \Gamma \models A.$$

Come nel caso del sistema hilbertiano la correttezza e completezza forte si riducono alla correttezza e completezza debole.

Per la correttezza e completezza debole si può dare una dimostrazione diretta, oppure osservare che:

Correttezza: Per definizione di espansione.

Completezza: Gli schemi di assioma a, b e c hanno una tableau dimostrazione (si veda l'esercizio 114) e sia MP che SU conservano la tableau dimostrabilità (si veda l'esercizio 115). Quindi $\vdash_H$ implica $\vdash_T$.

4.4.1 Esercizi

Nota: Negli esercizi di questo gruppo (108 – 115) il simbolo $\vdash$ sta per $\vdash_T$.

Esercizio 108 *Dimostrare che, dato un insieme finito* Γ *di formule proposizionali, la costruzione di un tableau per* Γ *termina.*

Esercizio 109 *Dimostrare, mediante tableau, che dall'insieme di formule* $\Gamma = \{P \leftrightarrow Q \vee R, S \rightarrow \neg R, S \wedge P\}$ *si deriva* $Q \leftrightarrow P$.

Esercizio 110 *Dimostrare, mediante tableau che:*

1. $P \wedge Q \rightarrow P$;
2. $(P \vee Q) \leftrightarrow \neg(\neg P \wedge \neg Q)$;
3. $(P \wedge Q) \leftrightarrow \neg(\neg P \vee \neg Q)$;
4. $((A \rightarrow B) \wedge (B \rightarrow C)) \rightarrow (A \rightarrow C)$
5. $(P \rightarrow Q) \wedge (Q \rightarrow P) \rightarrow (P \leftrightarrow Q)$.

Esercizio 111 *Considerare i seguenti enunciati:*

1. $(A \leftrightarrow B) \leftrightarrow ((\neg A \vee \neg B) \rightarrow (\neg A \wedge \neg B))$;
2. $(A \rightarrow (B \vee C) \wedge \neg(\neg A \rightarrow (B \wedge C)) \wedge (\neg B \wedge \neg C))$;
3. $(\neg A \rightarrow C) \wedge (A \rightarrow C) \wedge (A \rightarrow \neg B \wedge \neg C)$;
4. $(A \rightarrow (B \rightarrow C) \leftrightarrow ((A \wedge \neg B) \vee (A \rightarrow C))$;
5. $(A \rightarrow C) \wedge (A \wedge \neg B) \wedge (A \rightarrow (B \wedge C)))$;
6. $(A \rightarrow C) \wedge (A \rightarrow \neg C) \wedge (A \rightarrow (\neg B \wedge C)))$.

Dimostrare, usando il metodo dei tableau, quale dei precedenti enunciati è un teorema, quale è refutabile e quale è soddisfacibile. Fornire un modello per gli enunciati soddisfacibili.

Esercizio 112 *Dimostrare le proprietà di inclusione, monotònia e taglio per i tableau.*

Esercizio 113 *Dimostrare che anche per il sistema dei tableau vale il teorema di deduzione.*

Esercizio 114 *Dimostrare che gli assiomi del sistema hilbertiano hanno una tableau-dimostrazione.*

Esercizio 115 *Dimostrare che da* A *e* $A \rightarrow B$ *si deriva* B *col metodo dei tableau. Dimostrare che se* $\Gamma \vdash F[p]$ *allora* $\Gamma \vdash F[X/p]$.

4.5 Forma normale a clausole

Introduciamo ora la forma normale a clausole e un algoritmo per convertire le formule del calcolo proposizionale in formule equivalenti espresse in forma normale a clausole. Questa ci servirà nel prossimo paragrafo per introdurre il metodo di risoluzione.

Ricordiamo che nel calcolo proposizionale i *letterali* sono simboli proposizionali (*atomi*) o simboli proposizionali negati (atomi negati).

Definizione 4.31 *Una* formula *del calcolo proposizionale* $\mathcal{L}$ *è detta in* forma normale negativa *se il segno di negazione compare solo davanti agli atomi.*

Definizione 4.32 *Una* clausola *del calcolo proposizionale* $\mathcal{L}$ *è una disgiunzione di letterali* $L_1 \vee L_2 \vee \cdots \vee L_n$.

Definizione 4.33 *Una clausola* C_2 *è* sussunta *da una clausola* C_1 *se ogni letterale in* C_1 *occorre in* C_2.

Si noti che se $\mathcal{M}$ è modello per una clausola C lo è anche per tutte le clausole da essa sussunte.

Definizione 4.34 *Una* formula *del calcolo proposizionale* $\mathcal{L}$ *è detta in* forma normale congiuntiva *o in* forma clausale *oppure si dice che essa è un* insieme di clausole *se è in forma normale negativa e inoltre ha la forma* $C_1 \wedge C_2 \wedge \cdots \wedge C_n$ *dove le* C_i *sono clausole.*

Poiché la disgiunzione è commutativa, una clausola generica può essere scritta come:

$$A_1 \vee A_2 \vee \cdots \vee A_n \vee \neg B_1 \vee \neg B_2 \vee \cdots \vee \neg B_m \tag{4.2}$$

dove gli A_i e B_j sono atomi. Se $n = m = 0$ si ha la *clausola vuota* e si scrive $\{\}$, oppure $\Box$.

Una clausola verrà nel seguito anche indicata come l'insieme dei suoi letterali e cioè $\{L_1, L_2, \ldots, L_p\}$.
Se L è un letterale e C una clausola $C = \{L_1, L_2, \ldots, L_p\}$ talvolta scriveremo $L \cup C$ per indicare la clausola $C' = \{L\} \cup C$, cioè $C' = \{L, L_1, L_2, \ldots, L_p\}$.

Se $n = 1$, cioè se (4.2) ha la forma:

$$A_1 \vee \neg B_1 \vee \ldots, \vee \neg B_m$$

si parla di *clausola definita.*

Introduciamo ora un algoritmo per convertire le formule del calcolo proposizionale in forma a clausole (cioè congiunzione di disgiunzioni di letterali).

Utilizzando lo schema di decomposizione in α e β formule presentato nel paragrafo precedente, è possibile fornire un algoritmo non deterministico per trasformare una formula proposizionale F in un insieme di clausole $\{C_1, \ldots, C_n\}$.

Algoritmo 4.35 [TRASFORMAZIONE IN CLAUSOLE]
Sia F una formula proposizionale:

Passo 1 Si ponga $\{F\}$ come insieme iniziale.

Passo n+1 Si assuma di avere completato il passo n producendo un insieme $\{D_1, \ldots, D_n\}$, dove ciascun D_i è una disgiunzione della forma $\{A_1^i, \ldots, A_k^i\}$. Se qualche A_j^i non è un letterale, allora non si è raggiunta una forma normale congiuntiva e quindi:

Si scelga un elemento D_i che contiene qualche elemento che non è un letterale. Sia esso X:

a. se X è $\neg\top$ lo si sostituisca con $\bot$;
b. se X è $\neg\bot$ lo si sostituisca con $\top$;
c. se X è $\neg\neg A$ lo si sostituisca con A;
d. se X è una β formula si sostituisca con β_1, β_2;
e. se X è una α formula si sostituisca D_i con due clausole D_i^1 e D_i^2, dove D_i^1 è D_i in cui α è stata sostituita con α_1 e D_i^2 è D_i in cui α è stata sostituita con α_2.

L'algoritmo 4.35 trasforma un formula F in un insieme di clausole. Il lettore può dimostrare che esso termina (si veda l'esercizio 116).

Verifichiamo ora che il passo induttivo garantisce che la trasformazione indotta dalle regole a – e dell'algoritmo 4.35 è corretta, ovvero:

Lemma 4.36 *Sia D una disgiunzione della forma $\{A_1, \ldots, A_k\}$. Se D' è ottenuto da D applicando una delle regole a – e dell'algoritmo 4.35, abbiamo che $D \equiv D'$.*

Dimostrazione Sia $D = \{A_1, \ldots, X, \ldots, A_k\}$ dove X è $\neg\top$ o $\neg\bot$ o $\neg\neg A$. Applicando una delle regole a – c otteniamo $D' = \{A_1, \ldots, X', \ldots, A_k\}$. È facile verificare che $\vdash \neg\top \leftrightarrow \bot$, $\vdash \neg\bot \leftrightarrow \top$ e $\vdash \neg\neg A \leftrightarrow A$. Quindi, per il teorema 3.15, $D \equiv D'$.

Sia $D = \{A_1, \ldots, \alpha, \ldots, A_k\}$, applicando l'$\alpha$-regola otteniamo:

$$D_1 = \{A_1, \ldots, \alpha_1, \ldots, A_k\}$$

$$D_2 = \{A_1, \ldots, \alpha_2, \ldots, A_k\}.$$

Per mezzo delle regole di distribuzione è facile verificare che:

$$\{A_1, \ldots, \alpha_1, \ldots, A_k\} \wedge \{A_1, \ldots, \alpha_2, \ldots, A_k\} \equiv (D_1 \wedge D_2) \leftrightarrow (D' \leftrightarrow D).$$

Sia $D = \{A_1, \ldots, \beta, \ldots, A_k\}$. Applicando la β-regola otteniamo:

$$D' = \{A_1, \ldots, \beta_1, \beta_2, \ldots, A_k\}.$$

Per definizione di disgiunzione, $D \equiv D'$. □

Possiamo facilmente verificare, a questo punto, che partendo da una formula proposizionale qualunque F l'algoritmo 4.35 genera una sequenza di clausole $D_1, \ldots, D_m$ tali che la loro congiunzione è F in forma normale congiuntiva.

In realtà, data una formula proposizionale F in cui tutte le occorrenze di $\rightarrow$ e $\leftrightarrow$ sono state eliminate per mezzo di tautologia e la negazione è stata portata internamente applicando le leggi di De Morgan, sono sufficienti le regole di distribuzione per trasformare un enunciato F nell'enunciato F^C in forma normale congiuntiva:

Lemma 4.37 *Per ogni formula proposizionale F, in cui occorrono solo $\vee, \wedge$ e $\neg$, se F^C è la formula proposizionale ottenuta applicando le regole di distribuzione, si ha:*

$$\vdash F \leftrightarrow F^C$$

Dimostrazione Per induzione strutturale.
(*Passo Base*) Se F è un letterale o una clausola, è già in forma normale congiuntiva e, dunque $\vdash F \leftrightarrow F^C$.
(*Passo Induttivo*) Sia $F = \alpha_1 \wedge \alpha_2$, per ipotesi induttiva α_1 e α_2 sono in forma normale congiuntiva, ovvero, $\alpha_1^C = C_1^1 \wedge \ldots \wedge C_{k_1}^1$, e $\alpha_2^C = C_1^2 \wedge \ldots \wedge C_{k_2}^2$; inoltre $\vdash \alpha_1 \leftrightarrow \alpha_1^C$ e $\vdash \alpha_2 \leftrightarrow \alpha_2^C$. Pertanto, ponendo $F^C = \alpha_1^C \wedge \alpha_2^C$, F^C è ancora in forma a clausole, poiché è della forma $(C_1^1 \wedge \ldots \wedge C_{k_1}^1) \wedge (C_1^2 \wedge \ldots \wedge C_{k_2}^2)$. Ne consegue che $\vdash F \leftrightarrow F^C$.

Sia $F = \beta_1 \vee \beta_2$, per ipotesi induttiva β_i^C è della forma $C_1^i \wedge \ldots \wedge C_m^i$; inoltre $\vdash \beta_1 \leftrightarrow \beta_1^C$, $\vdash \beta_2 \leftrightarrow \beta_2^C$. Utilizzando le proprietà distributive del $\vee$ e del $\wedge$, descritte in 3.2, poniamo $F^C = (C_1^1 \vee C_1^2) \wedge \ldots \wedge (C_1^1 \vee C_{k_1}^2) \wedge \ldots \wedge (C_{k_1}^1 \vee C_{k_2}^2)$, chiaramente $F^C \leftrightarrow (\beta_1^C \vee \beta_2^C)$ e, inoltre, F^C è in forma normale congiuntiva. Ne segue che $\vdash F \leftrightarrow F^C$. □

Presentiamo i passi a – e dell'algoritmo in forma di regole di espansione:

$$1.\ \frac{\neg\neg A}{A} \quad 2.\ \frac{\neg\top}{\bot} \quad 3.\ \frac{\neg\bot}{\top} \quad 4.\ \frac{\alpha}{\alpha_1 \mid \alpha_2} \quad 5.\ \frac{\beta}{\beta_1, \beta_2} \tag{4.3}$$

Le regole in 4.3 sono duali rispetto alle regole di espansione per i tableau. Nel caso dei tableau si genera un albero, in quanto un albero può essere visto come la disgiunzione dei suoi rami che, a loro volta, possono essere visti come la congiunzione delle formule che etichettano i nodi. Nel caso delle regole 4.3, esse trasformano una formula in una sequenza di clausole, in altri termini l'applicazione di ciascuna regola crea una nuova sequenza di clausole da aggiungere alla precedente. Quindi, in sintesi, un tableau è una disgiunzione di congiunzioni, mentre una forma normale congiuntiva è una congiunzione di disgiunzioni.

Per maggiore chiarezza sul tipo di trasformazione, presentiamo un esempio.

Esempio 42 *Sia* $F = (P \to (Q \to (S \vee T))) \to (T \to Q)$. *Generiamo per passi la forma normale a clausole.*

1. $F = (P \to (Q \to (S \vee T))) \to (T \to Q)$
2. $C = \{\neg(P \to (Q \to (S \vee T))), (T \to Q)\}$
   (applicazione della β-regola a F)
3. $C_1 = \{P, (T \to Q)\}$, $C_2 = \{\neg(Q \to (S \vee T)), (T \to Q)\}$
   (applicazione della α-regola a C)
4. $C_1 = \{P, \neg T, Q\}$, $C_2 = \{\neg(Q \to (S \vee T)), (T \to Q)\}$
   (applicazione della β-regola a C_1)
5. $C_1 = \{P, \neg T, Q\}$, $C_2 = \{\neg(Q \to (S \vee T)), \neg T, Q\}$
   (applicazione della β-regola a C_2)
6. $C_1 = \{P, \neg T, Q)\}$, $C_2 = \{Q, \neg T, Q\}$, $C_3 = \{\neg(S \vee T), \neg T, Q\}$
   (applicazione della α-regola a C_2)
7. $C_1 = \{P, \neg T, Q\}$, $C_2 = \{Q, \neg T, Q\}$, $C_3 = \{\neg S, \neg T, Q\}$, $C_4 = \{\neg T, \neg T, Q\}$
   (applicazione della α-regola a C_3)

Notare che abbiamo introdotto nuovi pedici per gli insiemi C_i solo quando abbiamo applicato a essi la regola α. Come si può vedere dall'esempio 42, F è una formula soddisfacibile e la congiunzione delle clausole in 7 fornisce una formula F^C equivalente a F e, pertanto, anche essa soddisfacibile.

4.5.1 Esercizi

Esercizio 116 *Dimostrare che l'algoritmo 4.35 termina.*

Esercizio 117 *Dimostrare che, data una formula proposizionale F, l'algoritmo 4.35 trasforma F in una sequenza di clausole $D_1, \ldots, D_i, \ldots, D_m$, tali che*

$$\bigwedge\{D_1, \ldots, D_m\} \equiv F.$$

Usare il lemma 4.36 e il teorema 3.15.

Esercizio 118 *Riscrivere opportunamente l'algoritmo precedente utilizzando un ciclo while e le regole di riscrittura 4.3 sopra enunciate.*

Esercizio 119 *Trasformare in forma normale congiuntiva il seguente enunciato:*

$$(P \wedge (Q \vee R \vee (\neg S \wedge \neg Q)) \leftrightarrow (P \to (\neg S \vee (R \wedge Q)))).$$

4.6 Risoluzione nel caso proposizionale

Definizione 4.38 *Siano* $\{L\} \cup D_1$ *e* $\{\neg L\} \cup D_2$ *due clausole. Diciamo che la clausola* $D_1 \cup D_2$ *è ottenuta da* $\{L\} \cup D_1$ *e* $\{\neg L\} \cup D_2$ *per mezzo di* un passo di risoluzione *e scriviamo*

$$\frac{\{L\} \cup D_1 \quad \{\neg L\} \cup D_2}{D_1 \cup D_2} \tag{4.4}$$

Analogamente ai tableau, possiamo rappresentare la risoluzione come un albero binario. Diamo qui di seguito una definizione informale.

Un *albero di risoluzione* è un albero binario i cui nodi sono etichettati da clausole. Se C_1 e C_2 sono le etichette di due nodi fratelli il cui padre è etichettato da C, allora $C_1 = D_1 \cup \{L\}$, $C_2 = D_2 \cup \{\neg L\}$ e $C = D_1 \cup D_2$. In altri termini, l'etichetta associata al nodo padre è il risultato di un passo di risoluzione (un'applicazione della regola 4.4) applicato alle etichette dei due nodi figli.

Definizione 4.39 *Sia* Γ *un insieme finito di clausole, e* C *una clausola. Diciamo che* C si dimostra per risoluzione *da* Γ *sse esiste un albero di risoluzione la cui radice è etichettata da* C *e tutte le foglie dell'albero sono etichettate con clausole di* Γ. *Scriviamo* $\Gamma \vdash_R C$ *per indicare che* Γ *dimostra* C *per risoluzione.*

Esempio 43 *Sia* $\{\}$ *la clausola vuota e sia* $\Gamma = \{\{P\}, \{Q\}, \{\neg P, \neg Q\}\}$ *mostriamo l'albero di risoluzione per* $\Gamma \vdash_R \{\}$.

$$\dfrac{\dfrac{P \qquad \dfrac{Q \quad \neg P \vee \neg Q}{\neg P}}{\{\}}}{}$$

La risoluzione è una regola che conserva la soddisfacibilità, ovvero:

Teorema 4.40 [RISOLUZIONE E SODDISFACIBILITÀ]

1. *Sia* Γ *un insieme di clausole,* $\{L\} \cup D_1 \in \Gamma$ *e* $\{\neg L\} \cup D_2 \in \Gamma$. *Se* Γ *è soddisfacibile allora* $\Gamma \cup \{D_1 \cup D_2\}$ *è soddisfacibile;*
2. *Sia* Γ *un insieme di clausole, se* $\Gamma \vdash_R \{\}$ *allora* Γ *è insoddisfacibile.*

Dimostrazione Proviamo solo 1, poiché 2 ne è una diretta conseguenza. Supponiamo Γ soddisfacibile, cioè supponiamo che esista un $\mathcal{M}$ tale che $\mathcal{M} \models \Gamma$ e supponiamo che $\mathcal{M} \models L$; ne segue che $\mathcal{M} \not\models \neg L$ e quindi $\mathcal{M} \models D_2$. Ne segue che $\mathcal{M} \models D_1 \cup D_2$. Siccome per ipotesi $\mathcal{M} \models \Gamma$, ne segue che $\mathcal{M} \models \Gamma \cup \{D_1 \cup D_2\}$. Il caso in cui $\mathcal{M} \models \neg L$ è analogo. □

In effetti noi identifichiamo la clausola vuota $\{\}$ con la contraddizione, dunque la clausola vuota è insoddisfacibile. Siccome la regola di risoluzione conserva la soddisfacibilità, se una espansione di risoluzione contiene la clausola vuota questo significa che l'insieme di clausole di partenza non è soddisfacibile, ossia è contraddittorio.

Questa osservazione ci permette di usare la risoluzione, analogamente al sistema dei tableau, come un sistema di refutazione.
Quindi, per usare la risoluzione come metodo per dimostrare che una formula F è un teorema basta negare la formula, metterla in forma a clausole e vedere che, per risoluzione, essa genera la clausola vuota.

Esempio 44 *Costruiamo una refutazione per risoluzione per dimostrare che la formula $A \to C$ segue dalle due formule $A \to B$ e $B \to C$:*

1. $(A \to B) \wedge (B \to C) \wedge \neg(A \to C)$
2. $\{\neg A \vee B\}, \{\neg B \vee C\}, \{A\}, \{\neg C\}$ (trasformazione a clausole)
3. $\{\neg A \vee C\}, \{A\}, \{\neg C\}$(risolvendo B e $\neg B$ nella prima e seconda clausola di 2)
4. $\{C\}, \{\neg C\}$ (risolvendo $\neg A$ e A nella prima e seconda clausola di 3)
5. $\{\}$ (risolvendo C e $\neg C$ nella prima e seconda clausola di 4).

La risoluzione è un procedimento facilmente meccanizzabile, vista la sua uniformità; essa però è altamente inefficiente, a causa del nondeterminismo legato alla scelta delle clausole e dei letterali su cui effettuare passi di risoluzione. Sono state individuate strategie per aumentarne l'efficienza. A esse accenneremo nel paragrafo 8.4, quando tratteremo la risoluzione nella logica del primo ordine.

La *completezza* della risoluzione dovrebbe essere enunciata come segue:

$$\Gamma \vdash_R C \text{ sse } \Gamma \models C.$$

Tuttavia si consideri il seguente esempio: Sia $\Gamma = \{\{P\}\}$ e sia $C = \{P, Q\}$, è chiaro che C è una conseguenza logica di Γ ma non è vero che $\Gamma \vdash_R C$: non si può, infatti, applicare alcun passo di risoluzione.

Tuttavia, ciò che si può stabilire è il seguente teorema:

Teorema 4.41 [COMPLETEZZA DELLA RISOLUZIONE PER LA SODDISFACIBILITÀ] *Sia Γ un insieme di clausole. Γ è insoddisfacibile sse $\Gamma \vdash_R \{\}$.*

Anche in questo caso, come in quello dei tableau, per la correttezza e completezza si può dare una dimostrazione diretta, oppure osservare che:

> *Correttezza*: Ogni passo di risoluzione conserva la soddisfacibiltà (si veda il teorema 4.40).
>
> *Completezza*: Gli schemi di assioma a, b e c hanno una dimostrazione per risoluzione (si veda l'esercizio 122) e sia MP che SU conservano la dimostrabilità per risoluzione (si veda l'esercizio 123). Quindi $\vdash_H$ implica $\vdash_R$.

4.6.1 Esercizi

Nota: Negli esercizi di questo gruppo (120 – 125) il simbolo $\vdash$ sta per $\vdash_R$.

Esercizio 120 *Dimostrare le proprietà di inclusione, monotònia e taglio per la risoluzione.*

Esercizio 121 *Dimostrare che anche per la risoluzione vale il teorema di deduzione.*

Esercizio 122 *Dimostrare che gli assiomi del sistema hilbertiano hanno una dimostrazione per risoluzione.*

Esercizio 123 *Dimostrare che da A e $A \to B$ si deriva B con la risoluzione. Dimostrare che se $\Gamma \vdash F[p]$ allora $\Gamma \vdash F[X/p]$.*

Esercizio 124 *Dimostrare, usando la risoluzione, che:*

1. $P \wedge Q \to P$;
2. $(P \vee Q) \leftrightarrow \neg(\neg P \wedge \neg Q)$;
3. $(P \wedge Q) \leftrightarrow \neg(\neg P \vee \neg Q)$;
4. $((A \to B) \wedge (B \to C)) \to (A \to C)$;
5. $(P \to Q) \wedge (Q \to P) \to (P \leftrightarrow Q)$.

Esercizio 125 *Dati i seguenti enunciati:*

a. $P \leftrightarrow Q \leftrightarrow R$;

b. $(P \wedge (Q \wedge R)) \vee ((\neg A) \wedge ((\neg B) \wedge (\neg C)))$;

dimostrare mediante tableau o risoluzione che $\not\vdash a \to b$ e $\not\vdash b \to a$.

4.7 Riepilogo

In questo capitolo abbiamo presentato la *deduzione proposizionale*.

In particolare, abbiamo introdotto:

1. Un sistema assiomatico, cioè un insieme di assiomi e un apparato deduttivo per il calcolo proposizionale (*sistema hilbertiano*). Per esso abbiamo:
   (a) mostrato le proprietà di inclusione, monòtonia, compattezza e taglio sulle premesse;
   (b) presentato la dimostrazione del teorema di deduzione;
   (c) introdotto la nozione di insieme di formule consistente e insieme consistente massimale;
   (d) dato la dimostrazione dei teoremi di correttezza e completezza della deduzione rispetto alla conseguenza logica ($\vdash_H \equiv \models_H$).
2. La *deduzione naturale* proposizionale. Per essa abbiamo:
   (a) presentato le regole di introduzione ed eliminazione dei connettivi, del falso e la regola di riduzione all'assurdo;
   (b) illustrato la nozione di dimostrazione;
   (c) mostrato le proprietà di inclusione, monòtonia, compattezza e taglio sulle premesse;
   (d) presentato la dimostrazione del teorema di deduzione;
   (e) dato la dimostrazione dei teoremi di correttezza e completezza della deduzione rispetto alla conseguenza logica ($\vdash_{DN} \equiv \models_{DN}$).
3. Il *metodo dei tableau* proposizionali. Per fare ciò abbiamo:
   (a) introdotto le formule di tipo congiuntivo α e disgiuntivo β.
   (b) introdotto le regole di espansione per le formule;
   (c) fornito la nozione di tableau e di tableau-refutazione;
   (d) fornito un algoritmo per costruire tableau-refutazioni;
   (e) mostrato la correttezza e completezza della tableau-refutazione rispetto alla insoddisfacibità ($\vdash_T \equiv \models_T$).
4. La *risoluzione* proposizionale. Per fare ciò abbiamo:
   (a) introdotto la forma normale congiuntiva per le formule proposizionali;
   (b) fornito un algoritmo per trasformare le formule proposizionali in forma normale congiuntiva;
   (c) fornito la nozione di refutazione per risoluzione;
   (d) mostrato la correttezza e completezza della risoluzione rispetto alla insoddisfacibità ($\vdash_R \equiv \models_R$).

Capitolo 5
Teorie proposizionali

Questo capitolo sarà scritto nella prossima versione del testo. Nella presente versione le teorie vengono illustrate solo per la logica del primo ordine, nel capitolo 9.

Capitolo 6
Logica del primo ordine

Il linguaggio della logica proposizionale, presentato nei precedenti capitoli, ha un potere espressivo molto limitato. Infatti, il linguaggio proposizionale esprime, tramite un enunciato, solo le connessioni fra lettere proposizionali che a loro volta denotano, o rappresentano, fatti del mondo reale. Gli enunciati proposizionali sono indipendenti dalle particolari caratteristiche degli oggetti e individui di cui tali fatti trattano. L'enunciato $P \rightarrow M$ tratta l'implicazione e non le caratteristiche di ciò che P e M denotano, come ad esempio "Se un numero è pari allora è un multiplo di due".

I linguaggi del primo ordine, che introduciamo in questo capitolo, benché anche essi non adeguati per tutte le esigenze di rappresentazione del mondo reale, sono molto più espressivi del linguaggio proposizionale. Tramite un linguaggio del primo ordine si ha infatti la possibilità di esprimere proprietà e operazioni relative a individui e, quindi, si ha la possibilità di *predicare*[1] delle caratteristiche di oggetti e individui e, tramite l'uso di variabili e quantificatori, si può astrarre una caratteristica, o una operazione, da un individuo a una classe di individui. Si ha, infatti, nel linguaggio formale ciò che, nel linguaggio naturale, è espresso da "un", "alcuni" e "tutti", che permettono di indicare in modo generico oggetti che godono di proprietà.

Analogamente al caso proposizionale, insieme al linguaggio introduciamo una semantica che permette di interpretare i simboli del linguaggio. La semantica del primo ordine ci permetterà di capirne meglio la maggiore espressività rispetto al calcolo proposizionale.

[1] Di qui il nome *calcolo dei predicati*.

6.1 Linguaggi del primo ordine

Un *linguaggio del primo ordine* $\mathcal{L}$ è costruito sui seguenti insiemi di simboli:

Simboli Logici

- I connettivi proposizionali: $\neg$,$\wedge$,$\vee$, $\rightarrow$ e $\leftrightarrow$;
- Le costanti proposizionali $\top$ e $\bot$;
- Il simbolo di uguaglianza $=$, eventualmente assente;
- I simboli separatori '(', ')' e ',';
- Un'infinità numerabile di simboli di *variabile individuale* x_1, x_2, ...;
- Il simbolo di *quantificazione universale* $\forall$;
- Il simbolo di *quantificazione esistenziale* $\exists$.

Parametri

- Un insieme finito o numerabile di *simboli di predicato*, ognuno dei quali ha associato un intero positivo n detto arità. Un predicato di arità n è detto n-ario;
- Un insieme finito o numerabile di *simboli di funzione*, ognuno dei quali ha associato un intero positivo detto arità. Una funzione di arità n è detta n-aria;
- Un insieme finito o numerabile di *simboli di costante*.

Come nel linguaggio proposizionale, altri connettivi possono essere definiti in termini di quelli presentati sopra. L'insieme dei connettivi che abbiamo introdotto non è minimale, per esempio, come si vedrà nel seguito, il simbolo di quantificazione esistenziale $\exists$ può essere definito in termini di quello universale: $\exists = \neg\forall\neg$.

Osserviamo che i connettivi proposizionali, le variabili e le parentesi sono considerati simboli logici, in quanto sono gli elementi immutabili costituenti ogni linguaggio del primo ordine. Gli altri simboli, cioè i simboli di predicato, di costante e di funzione, sono considerati parametri poiché possono essere diversi, a seconda del linguaggio. Molti testi preferiscono chiamare l'insieme dei parametri una "segnatura" del linguaggio, in quanto i simboli della segnatura contraddistinguono il linguaggio stesso.

Si osservi che gli insiemi dei simboli di costante e funzione possono essere vuoti; non sono quindi essenziali nella definizione di una logica del primo ordine; viceversa l'insieme dei simboli di predicato – nel caso in cui il simbolo di uguaglianza sia assente – deve essere non vuoto. L'insieme dei simboli di variabile è infinito: vogliamo infatti avere sempre a disposizione variabili "fresche" da utilizzare nelle formule.

Per le variabili, nel seguito, piuttosto che usare pedici, preferiamo usare le ultime lettere dell'alfabeto minuscole: x, y, w, z, t. Inoltre, indicheremo i simboli di

predicato con $P, Q, R, S, \ldots$, oppure con parole con iniziale maiuscola. Indicheremo i simboli di funzione con $f, g, h, \ldots$, oppure con parole con iniziale minuscola. Ometteremo l'apice che denota l'arità quando questa sarà chiara dal contesto; infine, indicheremo le costanti con le lettere minuscole della prima parte dell'alfabeto oppure con nomi con iniziale minuscola.

Alcuni testi preferiscono non inserire i simboli di costante nell'alfabeto, inserendo al loro posto simboli di funzioni 0-arie. I simboli di predicato a un argomento, cioè di arità 1, vengono anche detti *monadici*.

Alcuni linguaggi del primo ordine includono tra i simboli logici il simbolo di *uguaglianza*, indicata con $=$. Il simbolo di uguaglianza è un simbolo di predicato binario che si distingue dagli altri simboli di predicato perché è un simbolo logico e ha un'interpretazione prefissata: qualunque sia il linguaggio del primo ordine e qualunque sia l'interpretazione che se ne dà, il simbolo di uguaglianza deve essere interpretato come l'identità sul dominio di interpretazione.

La ragione per la denominazione "primo ordine" sarà chiarita nel seguito.

Prima di iniziare l'esposizione delle regole di costruzione delle formule della logica, diamo qui di seguito alcuni esempi di linguaggi del primo ordine. Naturalmente, tutti condividono i simboli logici diversi dall'uguaglianza.

Esempio 45 *Il* linguaggio puro dei predicati*:*

Uguaglianza: assente;

Simboli di predicato n-ari: $P_1^n, P_2^n, \ldots$*;*

Simboli di costante: $c_1, c_2, \ldots$*;*

Simboli di funzione n-ari, $n > 0$*: nessuno.*

Esempio 46 *Il* linguaggio della teoria degli insiemi*:*

Uguaglianza: presente;

Simboli di predicato: un simbolo di predicato binario $\in$*;*

Simboli di costante: $\emptyset$*;*

Simboli di funzione n-ari, $n > 0$*: nessuno.*

Esempio 47 *Il* linguaggio della teoria elementare dei numeri*:*

Uguaglianza: presente;

Simboli di predicato: un simbolo di predicato binario $<$*;*

Simboli di costante: 0*;*

Simboli di funzione: un simbolo di funzione unario s*, che sta per la funzione successore, e i simboli di funzione binari* $+$ *e* $\times$*, che stanno per l'addizione e la moltiplicazione, rispettivamente.*

Come già detto, un linguaggio del primo ordine è specificato dall'insieme dei suoi simboli di predicato, dall'insieme dei suoi simboli di funzione e di costante. Ovvero esso è specificato dai parametri.

Diamo qui di seguito alcuni esempi di rappresentazione di frasi in linguaggi del primo ordine. Se vogliamo rappresentare

$$\begin{array}{l} \textit{Alcuni medici sono arroganti;} \\ \textit{Qualche operaio è metalmeccanico;} \end{array} \tag{6.1}$$

useremo un linguaggio del primo ordine puro (si veda l'esempio 45) e scriveremo:

$$\begin{array}{ll} a) & \exists x(Medico(x) \wedge Arrogante(x)); \\ b) & \exists x(Operaio(x) \wedge Metalmeccanico(x)). \end{array} \tag{6.2}$$

Osserviamo che abbiamo usato il quantificatore esistenziale $\exists$ per rappresentare "alcuni medici" e "qualche operaio"

Notare che con la formalizzazione appena esposta, di fatto noi ci diciamo sicuri dell'esistenza di medici arroganti e di operai metalmeccanici. Diverso sarebbe il significato se formalizzassimo le due frasi di 6.1 come:

$$\begin{array}{ll} a') & \exists x(Medico(x) \rightarrow Arrogante(x)); \\ b') & \exists x(Operaio(x) \rightarrow Metalmeccanico(x)). \end{array} \tag{6.3}$$

In questo caso di fatto non asseriamo l'esistenza di un medico. Infatti la formula a' di 6.3, essendo del tipo $M \rightarrow A$ è vera anche quando la premessa è falsa. Analogo ragionamento vale per la formula b'. Quindi la differenza sostanziale tra la traduzione 6.2 e la 6.3, è che la prima asserisce l'esistenza di medici arroganti e di operai metalmeccanici, la seconda no. Questa osservazione troverà ulteriore approfondimento quando parleremo di semantica della logica del primo ordine.

Questo esempio mette in luce una forma di ambiguità del linguaggio naturale. Quale tra la prima e la seconda rappresentazione sia più accurata dipende dal contesto del discorso.

Supponiamo invece di voler rappresentare in logica del primo ordine la frase:

$$\textit{Tutti i pasticcieri sanno fare la pasta frolla.} \tag{6.4}$$

anche in questo caso useremo un linguaggio del primo ordine puro (si veda l'esempio 45) e scriveremo:

$$\forall x(Pasticciere(x) \rightarrow SaFare(x, pasta\text{-}frolla)). \tag{6.5}$$

Per rappresentare "tutti i pasticcieri" abbiamo usato il quantificatore universale $\forall$. Per quanto riguarda invece la formalizzazione di questa frase non sorgono ambiguità nella scelta dei connettivi. Infatti, l'utilizzo del connettivo $\wedge$ sembra del tutto inappropriato in quanto:

$$\forall x(Pasticciere(x) \wedge SaFare(x, pasta\text{-}frolla)) \tag{6.6}$$

starebbe a significare che tutti (gli esseri umani) sono pasticceri e sanno fare la pasta frolla, il che sicuramente non riflette quello che la frase in 6.4 significa. Nessuna delle due formule in 6.5 e 6.6 comunque asserisce l'esistenza di pasticceri: tutte le proprietà sono infatti vere sull'insieme vuoto. Anche su questo torneremo quando introdurremo la semantica della logica del primo ordine, dove vedremo che, per ovviare al fatto che anche la seconda formula di 6.6 possa essere vacuamente verificata, si assume che il dominio di intepretazione delle variabili sia non vuoto per definizione.

Supponiamo, ora, di volere rappresentare frasi del tipo:

$$\begin{array}{l} \textit{Qualunque oggetto appartiene a un insieme di oggetti.} \\ \textit{Se due insiemi di oggetti hanno gli stessi elementi, allora sono uguali.} \end{array} \tag{6.7}$$

Per rappresentare in un linguaggio del primo ordine le frasi in 6.7, notiamo che esse parlano di "insiemi", di "elementi di insiemi" e di "appartenenza", dunque il linguaggio del primo ordine presentato nell'esempio 46 è il più appropriato:

$$\begin{array}{ll} 1. & \forall x \exists y (x \in y) \\ 2. & \forall y \forall z (\forall x (x \in y \leftrightarrow x \in z) \rightarrow y = z) \end{array} \tag{6.8}$$

Osserviamo che abbiamo trattato sia gli insiemi che gli elementi che a essi appartengono come oggetti individuali. Questa formalizzazione quindi non sarà sempre appropriata.

Infine costruiamo alcune frasi in cui si trattano i numeri naturali.

$$\begin{array}{l} \textit{Due più due è uguale a quattro;} \\ \textit{Per ogni numero } x \textit{ e } y \textit{ la somma di } x \textit{ e del successore di } y \\ \quad \textit{è uguale al successore della somma di } x \textit{ e } y; \\ \textit{Ogni numero diverso da zero è successore di qualche numero.} \end{array} \tag{6.9}$$

Per tali frasi, usiamo il linguaggio introdotto nell'esempio 47:

$$\begin{array}{l} s(s(0)) + s(s(0)) = s(s(s(s(0)))) \\ \forall x \forall y ((x + s(y)) = s(x + y)) \\ \forall x (x \neq 0 \rightarrow \exists y (x = s(y))) \end{array} \tag{6.10}$$

dove $a \neq b$ è una abbreviazione per $\neg(a = b)$. Si osservi che in (6.10) si assume che le variabili denotino numeri.

Definiamo ora le *espressioni legali* del linguaggio $\mathcal{L}$, cioè le formule. Esse hanno quelle caratteristiche importanti che abbiamo già visto per il linguaggio proposizionale: si possono comporre dando luogo a nuove espressioni legali e sono un insieme induttivo. Nel caso di un linguaggio del primo ordine gli elementi dell'insieme si compongono utilizzando i simboli logici e i parametri del linguaggio.

Per definire le formule dobbiamo prima definire i *termini* e le *formule atomiche* o *atomi*.

Definizione 6.1 [TERMINI] *L'insieme* TERM *dei* termini *di* $\mathcal{L}$ *è l'insieme induttivo definito come segue:*

1. *Ogni simbolo di costante e di variabile è un termine;*
2. *Se* $t_1 \ldots t_n$ *sono termini e* f *è un simbolo di funzione n-aria,* $f(t_1, \ldots, t_n)$ *è un termine (detto* termine funzionale*).*

Definizione 6.2 [ATOMI] *L'insieme* ATOM *degli* atomi *o* formule atomiche *è definito induttivamente come segue:*

1. $\bot$ *e* $\top$ *sono atomi;*
2. *Se* t_1 *e* t_2 *sono termini allora* $t_1 = t_2$ *è un atomo;*
3. *Se* $t_1, \ldots, t_n$ *sono termini e* P *è un simbolo di predicato n-ario* $P(t_1, \ldots, t_n)$ *è un atomo.*

L'insieme FBF delle *formule ben formate* di $\mathcal{L}$ è l'insieme delle espressioni che possono essere costruite a partire dai termini e dagli atomi usando opportunamente i connettivi e i quantificatori. In genere chiameremo una formula ben formata semplicemente *formula*, o anche *enunciato*.

Definizione 6.3 [FBF] *L'insieme delle* formule *di* $\mathcal{L}$ *è l' insieme induttivo definito come segue:*

- *Ogni atomo è una formula;*
- *Se* A *è una formula* $\neg A$ *è una formula;*
- *Se* $\circ$ *è un connettivo binario,* A *e* B *due formule,* $A \circ B$ *è una formula;*
- *Se* A *è una formula,* x *una variabile,* $\forall x A$ *e* $\exists x A$ *sono formule.*

Denotiamo, nel seguito, le formule di $\mathcal{L}$ sia con le lettere dell'alfabeto greco $\alpha, \beta, \gamma, \phi, \psi, \ldots$, sia con lettere maiuscole della prima parte dell'alfabeto, A, B, C, ... Insiemi di formule verranno indicati con lettere greche maiuscole Γ, Δ o anche con alcune lettere maiuscole, tipo T.

La *precedenza* tra gli operatori logici è stabilita come segue:

$$\forall, \exists, \neg, \wedge, \vee, \rightarrow, \leftrightarrow$$

e, come nel caso proposizionale, si assume che tutti gli operatori associno a destra.

Esempio 48 *Si consideri il seguente enunciato senza parentesi:*

$$\forall x P(x) \rightarrow \exists y \exists z Q(y, z) \wedge \neg \forall x R(x)$$

introducendo le parentesi per segnare la precedenza fra gli operatori otteniamo:

$$(\forall x P(x)) \rightarrow ((\exists y (\exists z Q(y, z))) \wedge (\neg (\forall x R(x)))).$$

Inoltre quando uno stesso simbolo di quantificazione si ripete, invece di scrivere $\forall x \forall y$ possiamo scrivere $\forall xy$, analogamente quando si ripete il quantificatore esistenziale.

Concludiamo questo paragrafo con qualche altro esempio relativo ai linguaggi introdotti precedentemente negli esempi 45 – 47.

Esempio 49 *Sia $\mathcal{L}$ un linguaggio puro dei predicati. Il seguente enunciato espresso nella lingua italiana*

> Ognuno ama qualcuno e nessuno ama tutti, oppure qualcuno ama chiunque e qualcuno non ama nessuno

è rappresentato in $\mathcal{L}$ come:

$$(\forall x \exists y\ ama(x,y) \wedge \neg \exists x \forall y\ ama(x,y)) \vee (\exists x \forall y\ ama(x,y) \wedge \exists x \forall y \neg\ ama(x,y)).$$

Esempio 50 *Sia $\mathcal{L}$ il linguaggio della teoria degli insiemi. L'enunciato*

> Non esiste un insieme tale che tutti gli insiemi siano suoi elementi

è rappresentato in $\mathcal{L}$ come:

$$\neg \exists x \forall y (y \in x).$$

Osserviamo che le variabili x e y utilizzate, denotano insiemi.

Esempio 51 *Sia $\mathcal{L}$ il linguaggio della teoria elementare dei numeri. L'enunciato*

> Non esiste un numero più grande di tutti i numeri

è rappresentato in $\mathcal{L}$ come:

$$\neg \exists y \forall x (y > x).$$

6.1.1 Variabili libere e legate

In una formula atomica, per definizione, non occorrono quantificatori, ma possono comparire variabili.

Consideriamo le seguenti formule non atomiche di $\mathcal{L}$:

$$\begin{array}{ll} 1. & \forall x \exists y \exists z (Mamma(y,x) \wedge Babbo(z,x)) \\ 2. & Film(x) \wedge Bello(x) \end{array} \tag{6.11}$$

In 1 le tre variabili x, y e z sono quantificate. Possiamo parafrasare l'enunciato come segue "Ognuno ha un padre e una madre". Qualunque sia il discorso in cui esprimiamo questa frase, essa può essere vera o falsa[2], comunque è dotata di senso.

In 2, invece, la variabile x occorre *libera*, poiché non ci sono quantificatori che la legano. Possiamo parafrasare la formula 2 di 6.11 come "un film bello", ma non sappiamo di quale film si stia parlando. Per dare un senso alla formula 2, essa

[2] Falsa, per esempio se la interpretiamo nell'insieme degli essere umani vivi.

deve essere inserita in un discorso più ampio che, eventualmente, ci permetta di individuare di quale film si tratti, o che dica che la proprietà è vera per tutti i film, o che ne esiste qualcuno. Una variabile libera denota un individuo rispetto a un contesto. In un certo senso, in italiano, come in un linguaggio del primo ordine, una frase con variabili libere si può considerare come una parte costitutiva di una frase più ampia. Ad esempio "un film bello" può essere parte di una frase del tipo "Giuseppe mi ha detto che ieri ha visto un film bello al Warner Village, il titolo del film è 'Matrix' ". Vedremo meglio cosa questo significhi quando ci occuperemo della semantica di $\mathcal{L}$.

Veniamo ora alla definizione di variabili libere. Per far ciò definiamo prima l'insieme delle variabili che occorrono in un termine e in una formula atomica.

Definizione 6.4 *L'insieme* $var(t)$ *delle* variabili di un termine t *è definito come segue:*

i. $var(t) = \{t\}$, *se* t *è una variabile;*

ii. $var(t) = \emptyset$, *se* t *è una costante;*

iii. $var(f(t_1, \ldots, t_n)) = \bigcup_{i=1}^{n} var(t_i)$

Un termine si dice chiuso *– o anche* ground *– se non contiene variabili.*

Definizione 6.5 *L'insieme delle variabili in una formula atomica* $R(t_1, \ldots, t_n)$ *è:*

$$var(R(t_1, \ldots, t_n)) = \bigcup_{i=1}^{n} var(t_i)$$

Un atomo si dice chiuso *– o anche* ground *– se non contiene variabili.*

Ovviamente le variabili che occorrono nei termini e negli atomi possono solo essere libere, in quanto gli unici operatori che "legano" variabili sono i quantificatori.

Prima di fornire una definizione di occorrenza libera o legata di variabili in una formula, definiamo il campo d'azione di un quantificatore.

Definizione 6.6 *Nella formula* $\forall xA$, *il* campo d'azione *del quantificatore* $\forall x$ *è* A. *Nella formula* $\exists xA$, *il* campo d'azione *del quantificatore* $\exists x$ *è* A.

Un quantificatore $\forall x$, $\exists x$ "lega" le (eventuali) occorrenze libere della *variabile quantificata* x nel campo d'azione A. Si noti che la nozione di campo di azione è coerente con la convenzione che stabilisce le precedenze dei connettivi.

Esempio 52 *La formula:*

$$\forall x \forall y \neg A \wedge \forall z B \rightarrow \exists u C$$

si legge:

$$((\forall x(\forall y(\neg A))) \wedge (\forall z B)) \rightarrow (\exists u C).$$

Quindi, C *è il campo d'azione di* $\exists u$; B *è il campo d'azione di* $\forall z$; $\neg A$ *è il campo d'azione di* $\forall y$; $\forall y(\neg A)$ *è il campo d'azione di* $\forall x$.

È chiaro che qualunque lettura diversa da quella convenzionale va imposta tramite un appropriato uso delle parentesi.

Definiamo ora, ricorsivamente, cosa significa per una *variabile* x occorrere *libera in una formula*:

Definizione 6.7

1. *Se A è un atomo, x occorre libera in A se x occorre in A;*
2. *x occorre libera in $(\neg A)$ se x occorre libera in A;*
3. *x occorre libera in $(A \circ B)$ se x occorre libera in A o x occorre libera in B;*
4. *x occorre libera in $\forall z A$ (rispettivamente $\exists z A$) se x occorre libera in A e $x \neq z$.*

Si noti quindi che l'insieme delle variabili libere di $\forall x A$ e di $\exists x A$ è:

$$var(\forall x A) = var(\exists x A) = var(A) - \{x\}.$$

Definizione 6.8 *Una occorrenza di una variabile x, in una formula, si dice* vincolata *o* legata *se non è libera.*

Quindi, le occorrenze legate di una variabile x in una formula $\forall x A$ sono quella dopo il quantificatore $\forall$ e le eventuali occorrenze libere di x nella formula A; queste ultime sono dette *occorrenze proprie.*

Definizione 6.9 *Un* enunciato – *detto altrimenti* formula chiusa – *è una formula senza occorrenze libere di variabili.*

Esempio 53 *Consideriamo la formula:*

$$\forall x(P(x) \rightarrow Q(x, y))$$

La variabile x è nel campo di azione del quantificatore $\forall x$ ed è legata da $\forall x$, quindi ha due occorrenze (proprie) legate, mentre la variabile y ha una occorrenza libera.

Nel seguito chiameremo occorrenze di una variabile legata solo le occorrenze proprie. Osserviamo che una stessa variabile può occorrere sia libera che legata in una formula.

Esempio 54
Nella formula $\forall x(\exists y P(x, y) \rightarrow Q(x, y))$ la variabile x ha due occorrenze legate, mentre la variabile y ha una occorrenza libera (quella in $Q(x, y)$) e una legata.

Nella formula $\forall x(Q(x) \wedge \exists y R(y, x))$ la variabile x ha due occorrenze legate dal quantificatore $\forall x$, mentre la variabile y ha una occorrenza legata dal quantificatore $\exists y$.

Nella formula $\forall x(Q(x) \wedge \exists x R(x, x))$ la variabile x ha tre occorrenze legate, la prima, quella in $Q(x)$ dal quantificatore $\forall x$, le altre dal quantificatore $\exists x$.

Le variabili legate sono talvolta dette "mute", in quanto il nome usato per esse è irrilevante ai fini del significato della formula. Questo sarà più chiaro quando introdurremo la semantica delle formule con quantificatori, ma già da ora è facile convincersi che alle formule $\forall x P(x)$ e $\forall y P(y)$ vogliamo attribuire lo stesso significato, che dipenderà dal dominio di interpretazione e dal significato che daremo al simbolo di predicato P, ma non dai nomi x e y usati per le variabili.

Si osservi comunque che sostituzioni di una variabile legata al posto di un'altra in una formula quantificata, senza che ciò cambi il significato della formula, possono essere fatte solo se questo non cambia i legami, come il seguente esempio mostra.

Esempio 55
Nella formula $\forall x(Q(x) \wedge \exists y R(y,x))$, sostituendo tutte le occorrenze di y con t otteniamo: $\forall x(Q(x) \wedge \exists t R(t,x))$, a essa equivalente.
Nella formula $\forall x(Q(x) \wedge \exists y R(y,x))$, sostituendo tutte le occorrenze di y con x otteniamo: $\forall x(Q(x) \wedge \exists x R(x,x))$. Le due formule non sono equivalenti dal punto di vista della semantica, perché i quantificatori agiscono diversamente sulle variabili.

Su questo argomento torneremo nel paragrafo 6.2.1, dove parleremo di equivalenza semantica; è infatti l'equivalenza semantica a legittimare il fatto che due diverse scritture per una formula possano essere considerate identiche e siano intercambiabili.

6.1.2 Esercizi

Esercizio 126 *Si individuino le variabili con occorrenze libere e legate nelle seguenti formule:*

$$\forall x(\exists y P(x,y) \rightarrow \exists z(Q(y,z) \rightarrow R(x,y) \wedge P(x,y)));$$
$$(\forall x(\exists y P(x,y) \rightarrow \exists z(Q(y,z)))) \rightarrow R(x,y) \wedge P(x,y);$$
$$(\forall x(\exists y P(x,y))) \rightarrow (\exists z(Q(y,z) \rightarrow R(x,y) \wedge P(x,y))).$$

Esercizio 127 *Scrivere un elenco di almeno dieci enunciati in cui una o più proprietà sono vere per tutti gli individui di un dato insieme.*

Esercizio 128 *Rappresentare in un linguaggio puro del primo ordine i seguenti enunciati:*

> Se è vero che ogni avo di un avo di un individuo è anche avo dello stesso individuo, allora deve esistere una persona che non ha avi;
>
> Tutte le donne sono belle e qualche uomo è bello.

Esercizio 129 *Rappresentare in un linguaggio del primo ordine con uguaglianza il seguente enunciato:*

Il papà di Gennaro batte a scopone i papà di tutti i bambini del quartiere Santa Lucia.

Esercizio 130 *La definizione data nel paragrafo 6.1.1 di occorrenza di variabile libera in una formula fa un tacito uso del teorema di ricorsione cui abbiamo accennato nel primo capitolo. Proviamo, dunque a definire una funzione h sulle formule atomiche come segue:*

$$h(x) = \begin{cases} 1 & \textit{se la variabile } x \textit{ occorre in } \alpha \\ 0 & \textit{altrimenti} \end{cases}$$

estendere la funzione h alla funzione $\overline{h}$ definita su tutte le formule di $\mathcal{L}$, in modo tale che $\overline{h}(x) = 1$ sse x occorre libera in α. L'esistenza di un'unica funzione $\overline{h}$ che estende h è garantita sia dal teorema di ricorsione che dal fatto che l'insieme delle formule di $\mathcal{L}$ è un insieme induttivo.

6.2 Interpretazioni e modelli

Nella logica proposizionale abbiamo introdotto delle funzioni di assegnazione che associano un valore di verità alle lettere proposizionali e, quindi, ricorsivamente agli enunciati. La nozione di interpretazione nella logica proposizionale ci serve per sapere se la rappresentazione della realtà che abbiamo dato per mezzo degli enunciati è vera o falsa. Essa autorizza solo una rappresentazione superficiale in cui gli oggetti del linguaggio hanno un valore di verità, ma non è possibile determinare una corrispondenza tra oggetti sintattici e oggetti del mondo reale. Nella logica del primo ordine abbiamo invece la possibilità di interpretare i simboli del linguaggio per mezzo di strutture. Una struttura del primo ordine è un oggetto matematico che ci permette di tradurre formule, espresse nel linguaggio del primo ordine, in espressioni che hanno un significato specifico relativamente alla struttura, cioè la realtà che stiamo rappresentando. Per questo motivo una struttura è un oggetto formale più complesso di una interpretazione proposizionale, poiché è necessario individuare l'insieme degli oggetti su cui si quantifica e cosa denotano gli altri parametri del linguaggio, ovvero i predicati, le costanti e le funzioni.

Intuitivamente una struttura per $\mathcal{L}$ deve fornirci una funzione che assegni al quantificatore $\forall$ un insieme non vuoto di elementi. Questa funzione, è la funzione di interpretazione e l'insieme non vuoto di elementi è il dominio di tale interpretazione. Più formalmente, diciamo che:

Definizione 6.10 *Una* struttura per il linguaggio $\mathcal{L}$ *è una coppia* $\mathfrak{A}$[3] $= \langle D, I \rangle$ *dove:*

- *D è un insieme non vuoto chiamato* dominio *di $\mathfrak{A}$;*

[3] Il simbolo $\mathfrak{A}$, è la A maiuscola dell'alfabeto gotico.

- *I è una funzione chiamata* interpretazione. *I associa:*
  - *a ogni simbolo di costante c un elemento $c^I \in D$;*
  - *a ogni simbolo di funzione n-aria f una funzione $f^I : D^n \to D$;*
  - *a ogni simbolo di predicato n-ario P una relazione n-aria $P^I \subseteq D^n$.*

La definizione precedente ci dice che l'interpretazione I assegna un preciso significato a ciascun parametro di $\mathcal{L}$. In particolare, il simbolo c è un nome per l'elemento c^I, che è un individuo (o un punto) di D; la formula atomica $P(t_1, \ldots, t_n)$ denota la n-upla di individui in D che sono a loro volta denotati da $t_1, \ldots, t_n$ e che stanno fra loro nella relazione P^I; la funzione $f(t_1, \ldots, t_k)$ denota l'operazione f^I in D sull'n-upla di individui denotati da $t_1, \ldots, t_k$. Osserviamo che è richiesto che il dominio sia non vuoto e che la funzione f^I sia definita su tutto il dominio. Il dominio D è il riferimento per l'interpretazione dei quantificatori: $\forall x$ significa per ogni elemento di D; $\exists x$ significa esiste un elemento di D. Come ovvio, I non fornisce un'interpretazione per $=, \top$ e $\bot$ in quanto sono simboli logici e non parametri del linguaggio, quindi la loro interpretazione è fissata.

Esempio 56 *Supponiamo di avere il linguaggio puro dei predicati, presentato nell'esempio 45 e il seguente enunciato:*

$$\forall x \exists y P(x, y) \tag{6.12}$$

Sia D, l'insieme degli esseri umani e P^I = l'insieme delle coppie $\langle A, B \rangle$, tale che B è padre *di A. Allora l'enunciato*

Tutti gli esseri umani hanno un padre

è la corretta interpretazione dell'enunciato 6.12 che intuitivamente è vero. Se consideriamo l'interpretazione I', tale per cui $P^{I'}$ è l'insieme delle coppie $\langle A, B \rangle$, tale che B è madre *di A, abbiamo un enunciato analogo: "Tutti gli esseri umani hanno una madre".*

Sia ora D l'insieme dei numeri naturali e J una interpretazione tale che P^J sia interpretato come l'insieme delle coppie $\langle m, n \rangle$, tale che $m < n$, allora l'enunciato:

Per ogni numero naturale ne esiste uno maggiore

è la corretta interpretazione di 6.12 e l'enunciato è ancora intuitivamente vero.

Per potere caratterizzare il significato di *verità in una struttura* è necessario definire come vengono interpretate le variabili.

Definizione 6.11 *Sia $\mathcal{V}ar$ l'insieme delle variabili di un linguaggio del primo ordine $\mathcal{L}$, una* assegnazione, *o ambiente, o* stato delle variabili *η in una struttura $\mathfrak{A} = \langle D, I \rangle$ è una funzione dall'insieme delle variabili $\mathcal{V}ar$ all'insieme D:*

$$\eta : \mathcal{V}ar \mapsto D$$

.

Un'assegnazione η quindi è una maniera di associare un valore alle variabili del linguaggio $\mathcal{L}$.

A questo punto definiamo la nozione di *formula A vera in una struttura $\mathfrak{A}$*. Il significato intuitivo è che A è vera in una struttura $\mathfrak{A} = \langle D, I\rangle$ se e solo se è vera la traduzione di A in D ottenuta assegnando un valore alle variabili per mezzo di una assegnazione η.

Per dare una definizione precisa, è necessario prima estendere la funzione di interpretazione ai termini del linguaggio.

Definizione 6.12 *Sia $\mathfrak{A} = \langle D, I\rangle$ una struttura per $\mathcal{L}$ e sia η una assegnazione. Estendiamo tale assegnazione a una assegnazione $\overline{\eta} = \langle I, \eta\rangle$ sui termini, ricorsivamente come segue:*

- *Per ogni variabile x, $x^{I,\eta} = x^{\eta}$;*
- *Per ogni costante c, $c^{I,\eta} = c^{I}$;*
- *Se $t_1, \ldots, t_n$ sono termini ed f è una funzione n-aria, allora*

$$f(t_1, \ldots, t_n)^{I,\eta} = f^{I}(t_1^{I,\eta}, \ldots, t_n^{I,\eta}).$$

Osserviamo che l'esistenza di un'unica $\overline{\eta}$ è assicurata dal teorema di ricorsione, sulla base del fatto che l'insieme dei termini è un insieme induttivo generato liberamente.

Un'assegnazione $\overline{\eta}$ è quindi una maniera di associare un valore ai termini del linguaggio $\mathcal{L}$. La definizione 6.12 infatti associa un elemento del dominio D ad ogni termine del linguaggio. Se il termine è chiuso, ovviamente il suo valore non dipende dall'assegnazione η.

Esempio 57 *Sia $\mathcal{L}$ specificato da un simbolo di quantificatore $\forall$, da un simbolo di predicato binario $\leq$, un simbolo di funzione unaria s, un simbolo di funzione binaria $+$ e un simbolo di costante $\emptyset$. Definiamo $\mathfrak{A}$ come segue:*

$D = \mathbb{N}$

$\leq^I = \{\langle m, n\rangle | m \leq n\}$

s^I *è la funzione successore s, cioè* $s^I(n) = (n+1)$;

$+^I$ *è l'operazione di addizione;*

$\emptyset^I = 0$.

Più semplicemente possiamo presentare la struttura $\mathfrak{A}$ come

$$\mathfrak{A} = \langle \mathbb{N}, \langle \leq, s, +, 0\rangle\rangle \text{ o anche } \mathfrak{A} = \langle \mathbb{N}, \leq, s, +, 0\rangle.$$

Dato $s(s(0)) + x$, un termine di $\mathcal{L}$, se fissiamo l'interpretazione η, tale che $x^{\eta} = 9$, allora $(s(s(0)) + x)^{I,\eta} = 11$. Se consideriamo il solo termine $s(s(0))$ questo è un termine chiuso e, dunque, il suo significato – che è 2 – è indipendente dall'assegnazione η che abbiamo utilizzato per interpretare la x in $\mathfrak{A}$.

A questo punto, definita una funzione che dà significato (cioè un valore) alle variabili, estesa quest'ultima ai termini del linguaggio, rispetto a una struttura in cui vengono interpretati predicati e funzioni, possiamo associare a ogni formula un valore di verità, quindi definire la nozione di soddisfacibilità e validità.

Lo facciamo introducendo una relazione di soddisfacibilità $\models$ tra struttura e formula, relativamente a una assegnazione η. La soddisfacibilità di una formula ϕ, in una struttura $\mathfrak{A}$ e rispetto a una assegnazione η alle variabili, è denotato con:

$$\mathfrak{A}, \eta \models \phi \tag{6.13}$$

che, intuitivamente, dice che la struttura $\mathfrak{A}$ soddisfa ϕ sse è vera l'interpretazione di ϕ determinata da $\mathfrak{A}$ e in cui una variabile x – ovunque occorra libera in ϕ – è valutata come x^η.

Definiamo, dunque, la relazione di soddisfacibilità $\models$ induttivamente come segue (dove "non $\models$" è scritto come "$\not\models$"):

Definizione 6.13 [SODDISFACIBILITÀ DELLE FORMULE] *Sia $\mathfrak{A} = \langle D, I\rangle$ una struttura per il linguaggio $\mathcal{L}$ e η una assegnazione in $\mathfrak{A}$.*

1. $$(\mathfrak{A}, \eta) \models \top \text{ e } (\mathfrak{A}, \eta) \not\models \bot;$$
2. *Se A è una formula atomica del tipo $P(t_1, \ldots, t_n)$, allora*
$$(\mathfrak{A}, \eta) \models P(t_1, \ldots, t_n) \text{ sse } \langle t_1^{I,\eta} \ldots t_n^{I,\eta}\rangle \in P^I;$$
3. *se A è una formula atomica del tipo $t_1 = t_2$ allora*
$$(\mathfrak{A}, \eta) \models t_1 = t_2 \text{ sse } t_1^{I,\eta} = t_2^{I,\eta};$$
4. $(\mathfrak{A}, \eta) \models \neg A$ *sse* $(\mathfrak{A}, \eta) \not\models A$;
5. $(\mathfrak{A}, \eta) \models A \wedge B$ *sse* $(\mathfrak{A}, \eta) \models A$ *e* $(\mathfrak{A}, \eta) \models B$;
6. $(\mathfrak{A}, \eta) \models A \vee B$ *sse* $(\mathfrak{A}, \eta) \models A$ *oppure* $(\mathfrak{A}, \eta) \models B$;
7. $(\mathfrak{A}, \eta) \models (A \rightarrow B)$ *sse* $(\mathfrak{A}, \eta) \models A$ *implica che* $(\mathfrak{A}, \eta) \models B$;
8. $(\mathfrak{A}, \eta) \models (A \leftrightarrow B)$ *sse* $(\mathfrak{A}, \eta) \models A$ *e* $(\mathfrak{A}, \eta) \models B$ *oppure* $(\mathfrak{A}, \eta) \not\models A$ *e* $(\mathfrak{A}, \eta) \not\models B$;
9. $(\mathfrak{A}, \eta) \models \forall x A$ *sse per ogni* $d \in D$ *è verificato che* $\mathfrak{A} \models A(\eta[d/x])$;
10. $(\mathfrak{A}, \eta) \models \exists x A$ *sse esiste un* $d \in D$ *per cui è verificato che* $\mathfrak{A} \models A(\eta[d/x])$.

Dove l'assegnazione $\eta[d/x]$ indica la funzione che si comporta esattamente come η eccetto che sulla variabile x, dove assume il valore d:

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

Osserviamo che nel punto 3 della definizione 6.13 abbiamo usato il simbolo $=$ sia come simbolo logico in $t_1 = t_2$ sia per denotare l'identità sul dominio. Abbiamo fatto un abuso linguistico; in realtà avremmo dovuto, sin dall'inizio, denotare l'uguaglianza con un simbolo peculiare, del tipo $\approx$. Tuttavia, invece di appesantire la simbologia, ci affidiamo al contesto per individuare quando trattiamo l'uguaglianza (sintattica, come in $t_1 = t_2$) o l'identità (semantica, come in $t_1^{I,\eta} = t_2^{I,\eta}$).

Notiamo che, differentemente dal caso proposizionale, qui non abbiamo prima fornito una funzione di valutazione e poi una relazione di soddisfacibilità, ma abbiamo subito introdotto la relazione $\models$.

Esempio 58 *Analizziamo il caso delle due seguenti formule*[4]*:*

1. $\exists x(P(x) \wedge Q(x))$
2. $\exists x(P(x) \rightarrow Q(x))$

In base a quanto visto nella definizione precedente al punto 10, esse sono verificate in una struttura $\mathfrak{A}$ *sse esiste un* $d \in D$ *per il quale è verificato che* $\mathfrak{A} \models (P(x) \wedge Q(x))(\eta[d/x])$ *e* $\mathfrak{A} \models (P(x) \rightarrow Q(x))(\eta[d/x])$.

Nel caso della formula 1 questo vuol dire che esiste un $d \in D$ tale che $d \in P^I \cap Q^I$ (ricordiamo che D è per definizione non vuoto). Quindi i sottoinsiemi di D associati da I a P e Q, rispettivamente, sono non vuoti e hanno intersezione non vuota (essa contiene almeno d).

Nel caso della formula 2 questo vuol dire che esiste un $d \in D$ tale che $d \in \overline{P^I} \cup Q^I$. Quindi la formula $\exists x(P(x) \rightarrow Q(x))$ è resa vera addirittura da tutti gli elementi del dominio, se I associa P al sottoinsieme vuoto di D.

Quindi, mentre dal fatto che la formula 1 è verificata in una struttura consegue che il predicato che interpreta P ha una estensione[5] non vuota in quella struttura, dalla formula 2 non possiamo trarre la stessa conclusione.

Esempio 59 *Analizziamo le formule:*

3. $\forall x(P(x) \wedge Q(x))$
4. $\forall x(P(x) \rightarrow Q(x))$

In base alla definizione 6.13, punto 9, la formula 3 dice che sia P che Q hanno un'estensione non vuota, anzi entrambe coincidono con l'intero dominio D (ricordiamo che D è per definizione non vuoto); la 4 ci dice soltanto che l'estensione di P è contenuta in quella di Q, ma non esclude affatto il caso in cui l'estensione di P sia vuota, né che entrambi P e Q abbiano un'estensione vuota.

Nella definizione 6.13, per il punto 9 si ha che, se A è chiusa (cioè non contiene variabili libere), il suo valore di verità non dipende dal particolare stato delle sue variabili (o assegnazione) η. D'altro canto, è evidente che la soddisfacibilità di una formula ϕ con variabili libere, in una struttura $\mathfrak{A}$, dipende essenzialmente dall'assegnazione η

[4] Le formule discusse in questo esempio e nel prossimo hanno la stessa struttura di quelle in 6.2, 6.3, 6.5 e 6.6 a pagina 120.

[5] Ricordiamo che si dice *estensione* di un predicato l'insieme degli elementi che lo soddisfano.

alle variabili x di ϕ, mentre non importa come η interpreti le altre variabili. Questo è assicurato dal seguente teorema:

Teorema 6.14 *Sia ϕ una formula e $\mathfrak{A}$ una struttura del primo ordine. Siano η_1 e η_2 due assegnazioni alle variabili che concordano su tutte le variabili che occorrono libere in ϕ allora:*

$$\mathfrak{A}, \eta_1 \models \phi \text{ sse } \mathfrak{A}, \eta_2 \models \phi.$$

Possiamo, dunque, dire in generale che una formula A di $\mathcal{L}$ è *soddisfacibile* se esiste una struttura $\mathfrak{A} = \langle D, I \rangle$ e una assegnazione alle variabili η tale che $(\mathfrak{A}, \eta) \models A$.

Il teorema 6.14 può essere generalizzato come segue:

Teorema 6.15 *Siano $\mathfrak{A}$ e $\mathfrak{B}$ due strutture che hanno lo stesso dominio e che concordano nell'interpretazione di tutti i parametri che occorrono in ϕ. Allora*

$$\mathfrak{A}, \eta \models \phi \text{ sse } \mathfrak{B}, \eta \models \phi.$$

Dimostrazione Per induzione su ϕ. Sia $\mathfrak{A} = \langle D, I \rangle$ e $\mathfrak{B} = \langle D, I' \rangle$.
(*Passo Base*) Sia ϕ una formula atomica del tipo $P(t_1, \ldots, t_n)$ e $\mathfrak{A}, \eta \models P(t_1, \ldots, t_n)$ allora $\langle t_1^{I,\eta}, \ldots, t_n^{I,\eta} \rangle \in P^I$. Per ipotesi, per ogni n-upla $\langle s_1, \ldots s_n \rangle \in D^n$, $\langle s_1, \ldots s_n \rangle \in P^I$ sse $\langle s_1, \ldots s_n \rangle \in P^{I'}$. Inoltre $t_i^{I,\eta} = t_i^{I',\eta}, 1 \leq i \leq n$, per ipotesi; dal teorema 6.14 consegue che $\mathfrak{B}, \eta \models P(t_1, \ldots, t_n)$.
(*Passo Induttivo*) Sia ϕ del tipo $\alpha \circ \beta$ (con $\circ$ connettivo binario), oppure del tipo $\neg\alpha$, per questi casi è immediato applicare l'ipotesi induttiva.
Sia ϕ del tipo $\forall x \psi$, allora le variabili libere di ϕ sono quelle di ψ, eccetto la x. E, per ipotesi induttiva $\mathfrak{A}, \eta \models \psi$ sse $\mathfrak{B}, \eta \models \psi$. Siccome $\mathfrak{A}$ e $\mathfrak{B}$ hanno lo stesso dominio, abbiamo che $\eta[d/x]$ si comporta nello stesso modo sulle due strutture; dal teorema 6.14 consegue che $\mathfrak{A}, \eta \models \forall x \psi$ sse $\mathfrak{B}, \eta \models \forall x \psi$.

Il caso del quantificatore esistenziale è analogo. □

Esempio 60 *Consideriamo la struttura definita nell'esempio 57 e la formula:*

$$\forall x \forall y (P(x, y) \rightarrow \exists z P(y, z))$$

Se l'interpretazione di P è $\leq$, la formula ci dice che per ogni coppia di numeri $\langle x, y \rangle$ con $x \leq y$ esiste un numero z tale che $y \leq z$. È chiaro che per verificare la soddisfacibilità di questa formula in $\mathfrak{A}$, comunque fissiamo una assegnazione alle variabili libere, dobbiamo considerare tutti gli elementi del dominio $D = \mathbb{N}$; siccome non ci sono variabili libere nella formula, per il teorema 6.14 l'assegnazione η è ininfluente.

In virtù dei precedenti teoremi abbiamo che la soddisfacibilità per le formule chiuse è effettivamente indipendente dall'assegnazione alle variabili libere. In altri termini, si può verificare che: $\mathfrak{A}, \eta \models \phi$, dove ϕ è una formula chiusa, o per ogni assegnazione η oppure per nessuna. Pertanto, data una formula chiusa ϕ, o ϕ è vera in $\mathfrak{A}$ oppure è falsa. Questo non vale per le formule aperte.

Se $\mathcal{L}$ è un linguaggio del primo ordine, possiamo dare le seguenti definizioni.

Definizione 6.16 *Se per una formula $A \in \mathcal{L}$, $(\mathfrak{A}, \eta) \models A$ è verificato per ogni assegnazione alle variabili, allora scriviamo $\mathfrak{A} \models A$ e diciamo che $\mathfrak{A}$ è un* modello *di A, ovvero che A è* vera *in $\mathfrak{A}$.*

Definizione 6.17 *Una formula $A \in \mathcal{L}$ è* valida *sse è vera in tutte le strutture di $\mathcal{L}$ e lo scriviamo $\models A$.*

Possiamo estendere quanto detto sopra a un insieme qualunque di formule di $\mathcal{L}$.

Definizione 6.18 *Un insieme di formule Γ è* soddisfacibile *se esiste una struttura $\mathfrak{A}$ ed una assegnazione η tale che $(\mathfrak{A}, \eta) \models A$ per ogni $A \in \Gamma$.*

Come abbiamo osservato precedentemente, per le formule chiuse la soddisfacibilità non dipende dalle assegnazioni, quindi la soddisfacibilità per una formula chiusa A coincide con la verità.

Definizione 6.19 *Sia Γ un insieme di formule e A una formula. Allora Γ* implica logicamente *A, scritto $\Gamma \models A$, sse per ogni struttura $\mathfrak{A}$ del linguaggio e ogni assegnazione η alle variabili, tale che $(\mathfrak{A}, \eta) \models B$ per ogni $B \in \Gamma$, è verificato che $(\mathfrak{A}, \eta) \models A$.*

La precedente definizione di implicazione logica, relativamente a formule, non è l'unica che si possa dare. In verità ce ne è un'altra, ovvero:
Sia Γ un insieme di formule e A una formula. Allora Γ implica logicamente *A, scritto $\Gamma \models A$, sse per ogni struttura $\mathfrak{A}$ per il linguaggio e ogni assegnazione η alle variabili, tale che $(\mathfrak{A}, \eta) \models B$ per ogni $B \in \Gamma$, è verificato che per ogni assegnazione η', $(\mathfrak{A}, \eta') \models A$.*
Noi adottiamo la 6.19. La differenza è significativa, principalmente in relazione al teorema di deduzione che vedremo nel prossimo capitolo.

Per le formule chiuse, siccome l'assegnazione η è ininfluente, le due definizioni coincidono. Ovvero, se A è un enunciato, $\Gamma \models A$ sse ogni modello di Γ è un modello di A.

Dato un dominio e un insieme di costanti, funzioni e predicati su di esso, li possiamo considerare una struttura per intepretare i parametri di un linguaggio del primo ordine. Un modo ovvio per costruire un tale linguaggio è usare una segnatura in cui ci sia esattamente un simbolo per ogni costante, funzione e predicato della struttura.

Ad esempio alla struttura

$$\langle D, R_1^I, \ldots, R_n^I, f_1^I, \ldots, f_m^I, c_1^I, \ldots, c_k^I \rangle$$

associamo il linguaggio del primo ordine la cui segnatura è

$$\langle R_1, \ldots, R_n, f_1, \ldots, f_m, c_1, \ldots, c_k \rangle.$$

Indichiamo questo linguaggio con $\mathcal{L}_{\mathfrak{A}}$. In altre parole, $\mathcal{L}_{\mathfrak{A}}$ è il linguaggio specificato dall'insieme dei parametri che sono interpretati nella struttura $\mathfrak{A}$: per ogni relazione n- aria R_i^I in $\mathcal{L}_{\mathfrak{A}}$ c'è un simbolo di predicato n-ario R_i, e analogamente c'è nel linguaggio un simbolo per ciascuna funzione e costante della struttura.

Esempio 61 *Una struttura* $\mathfrak{A} = \langle \mathbb{N}, +, \times, 0 \rangle$ *ha associato un linguaggio* $\mathcal{L}_{\mathfrak{A}}$ *in cui occorrono due simboli di funzione binari, per denotare l'addizione e la moltiplicazione rispettivamente, e un simbolo di costante per denotare lo zero.*

6.2.1 Equivalenza semantica

L'equivalenza semantica nella logica del primo ordine si definisce come segue.

Definizione 6.20 *Due formule* P *e* Q *si dicono* semanticamente (*o* logicamente) equivalenti *se per tutte le strutture* $\mathfrak{A}$ *e tutte le assegnazioni* η *si ha che:*

$$\mathfrak{A}, \eta \models P \text{ sse } \mathfrak{A}, \eta \models Q.$$

In questo caso si scrive $P \equiv Q$.

Sulla base della definizione di equivalenza semantica si può dimostrare che due formule che differiscono solo per il nome delle variabili mute sono semanticamente equivalenti. Per dimostrare ciò introduciamo il seguente lemma di sostituzione.

Lemma 6.21

1. *Se* y *non occorre in* t *allora* $t^{I,\eta[d/x]} = t[y/x]^{I,\eta[d/y]}$
2. *Se* y *non occorre in* P *allora per tutte le interpretazioni* $\mathfrak{A}$ *e tutte le assegnazioni* η *si ha che:*

$$\mathfrak{A}, \eta[d/x] \models P \text{ sse } \mathfrak{A}, \eta[d/y] \models P[y/x].$$

Dimostrazione La dimostrazione si effettua mediante induzione strutturale sui termini, per il caso 1 e sulle formule, per il caso 2. Essa è lasciata al lettore (si veda l'esercizio 143). □

Teorema 6.22 *Se* y *non occorre in* P

1. $\exists x P \equiv \exists y P[y/x]$
2. $\forall x P \equiv \forall y P[y/x]$.

Dimostrazione Segue immediatamente dal lemma 6.21. □

A questo punto possiamo rendere più precise le correlazioni tra il quantificatore universale ed esistenziale.

Teorema 6.23

1. $\forall x P \equiv \neg \exists x \neg P$

2. $\neg\forall x P \equiv \exists x \neg P$
3. $\exists x P \equiv \neg\forall x \neg P$
4. $\neg\exists x P \equiv \forall x \neg P$.

Dimostrazione Tutte le equivalenze su esposte sono conseguenze immediate delle definizioni. Dimostriamo la 1. Dobbiamo dimostrare che $(\mathfrak{A}, \eta) \models \forall x P$ sse $(\mathfrak{A}, \eta) \models \neg\exists x \neg P$. Ma $(\mathfrak{A}, \eta) \models \forall x P$, per definizione, vuol dire che per ogni $d \in D$, dominio di $\mathfrak{A}$ si ha che $\mathfrak{A} \models \forall x P[d/x]$; questo vuol dire che per nessun $d \in D$ $(\mathfrak{A}, \eta) \models \neg P[d/x]$, quindi $(\mathfrak{A}, \eta) \not\models \exists x \neg P[d/x]$; quindi $(\mathfrak{A}, \eta) \models \neg\exists x \neg P[d/x]$.
Gli altri casi sono lasciati al lettore (si veda l'esercizio 144). □

Analogamente si può dimostrare che l'ordine dei quantificatori dello stesso tipo è irrilevante ai fini del significato di una formula e che un quantificatore può essere cancellato se la variabile da esso quantificata non occorre libera nel suo campo d'azione:

Teorema 6.24

1. $\forall x \forall y P \equiv \forall y \forall x P$
2. $\exists x \exists y P \equiv \exists y \exists x P$
3. $\forall x P \equiv P$ *se* $x \notin var(P)$
4. $\exists x P \equiv P$ *se* $x \notin var(P)$.

Dimostrazione Lasciata al lettore (si veda esercizio 145). □

Osserviamo che i quantificatori sono distributivi rispetto ai connettivi $\wedge$ e $\vee$, ma con alcune restrizioni:

Teorema 6.25

1. $\forall x(P_1 \wedge P_2) \equiv \forall x P_1 \wedge \forall x P_2$
2. $\exists x(P_1 \vee P_2) \equiv \exists x P_1 \vee \exists x P_2$
3. $\forall x(P_1 \vee P_2) \equiv \forall x P_1 \vee P_2$ *se* $x \notin var(P_2)$
4. $\exists x(P_1 \wedge P_2) \equiv \exists x P_1 \wedge P_2$ *se* $x \notin var(P_2)$.

Dimostrazione Lasciata al lettore (si veda esercizio 146). □

Osserviamo infine che le condizioni $x \notin var(P_2)$, nei casi 3 e 4 del precedente teorema sono essenziali infatti:

Esempio 62 *Si può verificare che*

1. $\forall x(P_1 \vee P_2) \not\equiv \forall x P_1 \vee \forall x P_2$
2. $\exists x(P_1 \wedge P_2) \not\equiv \exists x P_1 \wedge \exists x P_2$

Un facile controesempio per provare che $\forall x(P_1 \vee P_2)$ non è semanticamente equivalente a $\forall x P_1 \vee \forall x P_2$ è il seguente: dire "Tutti i numeri naturali sono minori o maggiori di 100" è vero, ed è ben diverso rispetto a dire "Tutti i numeri naturali sono minori di 100, oppure tutti i numeri naturali sono maggiori di 100".

Un controesempio per provare che $\exists x(P_1 \wedge P_2)$ non è logicamente equivalente a $\exists x P_1 \wedge \exists x P_2$ è il seguente: dire che "Esiste un numero pari ed esiste un numero dispari" è ben diverso dal dire che "Esiste un numero che è pari e dispari".

Indichiamo genericamente con $\mathcal{Q}$ un quantificatore: $\mathcal{Q} \in \{\forall, \exists\}$. Sui quantificatori valgono le seguenti equivalenze.

Teorema 6.26

1. $\mathcal{Q}_1 x(P_1 \vee \mathcal{Q}_2 x P_2) \equiv \mathcal{Q}_1 x \mathcal{Q}_2 z(P_1 \vee P_2[z/x])$
2. $\mathcal{Q}_1 x(P_1 \wedge \mathcal{Q}_2 x P_2) \equiv \mathcal{Q}_1 x \mathcal{Q}_2 z(P_1 \wedge P_2[z/x])$

se $z \notin var(P_1) \cup var(P_2)$.

Dimostrazione Lasciata al lettore (si veda l'esercizio 147). Si noti che la condizione $z \notin var(P_1) \cup var(P_2)$ può essere sempre verificata, previa una opportuna ridenominazione delle variabili. □

Presentiamo ora le equivalenze che illustrano le relazioni tra i quantificatori e il connettivo di implicazione.

Teorema 6.27 *Sia* P_2 *una formula in cui* x *non occorre libera*

1. $\forall x P_1 \to P_2 \equiv \exists x(P_1 \to P_2)$
2. $\exists x P_1 \to P_2 \equiv \forall x(P_1 \to P_2)$
3. $P_2 \to \forall x P_1 \equiv \forall x(P_2 \to P_1)$
4. $P_2 \to \exists x P_1 \equiv \exists x(P_2 \to P_1)$.

Dimostrazione Lasciata al lettore (si veda l'esercizio 148). □

Esempio 63 *La formula*

$$\exists x(P(x) \to \forall x P(x))$$

è valida[6].

[6] La validità della formula dell'esempio 63 può sembrare controintuitiva, in quanto una lettura per essa è: "Esiste un uomo che, se porta il cappello lui, portano il cappello tutti", che è paradossale. Infatti questa formula è anche nota come "il paradosso del cappello". Ma se ci fidiamo delle dimostrazioni *dobbiamo* credere che è valida!

Infatti $\exists x(P(x) \to \forall x P(x))$ è logicamente equivalente a $\exists x(P(x) \to \forall y P(y))$ che, per il punto 1 del teorema 6.27, è logicamente equivalente a

$$\forall x P(x) \to \forall y P(y)$$

che è istanza della tautologia $A \to A$.

Un altro modo per convincersene è il seguente: $\exists x(P(x) \to \forall y P(y))$ è logicamente equivalente a $\exists x(\neg P(x) \vee \forall y P(y))$ e l'esistenziale distribuisce sull'or (punto 2 del teorema 6.25), per cui è logicamente equivalente a $\exists x \neg P(x) \vee \exists x \forall y P(y))$ che è logicamente equivalente a: $\neg \forall x P(x) \vee \forall y P(y))$, istanza della tautologia $\neg A \vee A$.

6.2.2 Esercizi

Esercizio 131 *Fornire un enunciato interpretato nella struttura dell'esempio 57.*

Esercizio 132 *Dimostrare il teorema 6.14, per induzione, usando la definizione ricorsiva di soddisfacibilità.*

Esercizio 133 *Fornire tre esempi di formule con variabili libere che non sono né vere né false in una data struttura* $\mathfrak{A}$.

Esercizio 134 *Verificare le seguenti:*

1. $\forall x Q(x) \models Q(y)$;
2. $Q(x) \not\models \forall y Q(y)$;
3. $\forall x Q(x) \models \exists y Q(y)$;
4. $\exists x \forall y P(x,y) \models \forall y \exists x P(x,y)$;
5. $\forall y \exists x P(x,y) \not\models \exists x \forall y P(x,y)$;
6. $\models \forall x(Q(x) \to \exists x Q(x))$.

Esercizio 135 *Considerare la struttura dell'esempio 57 e verificare se*

$$(\mathfrak{A}, \eta) \models \forall y P(y,x)$$

nel caso in cui P *è* $\leq$ *e* $x^\eta = 0$.

Esercizio 136 *Rappresentare la seguente* grammatica *come un insieme di formule del primo ordine:*

$$\begin{array}{lll} S & ::= & PN\ PV \\ PN & ::= & ART\ N \mid PN \\ PV & ::= & VT\ NP \mid VI \\ N & ::= & ragazzo|ragazza \\ NP & ::= & Francesco|Giovanna \\ ART & ::= & un|uno|il|lo|la \\ VT & ::= & ama \\ VI & ::= & ride \end{array}$$

Sia Γ l'insieme di formule che rappresentano al primo ordine la grammatica. Sia ϕ l'enunciato "il ragazzo ama Giovanna". Dimostrare che $\Gamma \models \phi$.

Esercizio 137 *Sia data una struttura $\mathfrak{A} = \langle \mathbb{N}, +, \times, 0 \rangle$ e la formula $\psi = \exists z(x \times z = y)$ nel linguaggio del primo ordine, con uguaglianza, $\mathcal{L}_{\mathfrak{A}}$. Consideriamo la sequenza di elementi $\{d_k\}_{k \in \mathbb{N}}$, di D, definita da $d_k = k + 2$. Verificare se tale sequenza di elementi soddisfa o meno ψ in $\mathfrak{A}$.*

Esercizio 138 *Considerare il seguente insieme di formule Σ[7]:*

1. $\forall x(Canarino(x) \rightarrow Uccello(x))$
2. $\forall x(Struzzo(x) \rightarrow Uccello(x))$
3. $\forall x(Passero(x) \rightarrow Uccello(x)$
4. $\forall x(Uccello(x) \wedge \neg Eccezione(x) \rightarrow Vola(x))$
5. $Canarino(titti) \wedge Struzzo(fred) \wedge Passero(alfredo)$

Sapendo che gli struzzi sono gli unici a fare eccezione, verificare se

1. $\Sigma \models Vola(alfredo)$
2. $\Sigma \models \neg Vola(titti)$
3 $\Sigma \models \neg(Vola(titti) \wedge Vola(fred))$

Rappresentare i modelli di Σ. Suggerimento: si usi come dominio per interpretare i parametri l'insieme dei parametri stessi. Cioè

$$\mathfrak{A} = \langle \{alfredo, titti, fred\}, Vola, Uccello, Canarino \ldots \rangle.$$

Esercizio 139 *Sia Γ un insieme di formule e ϕ una formula. Dire se nel caso in cui $\Gamma \not\models \phi$ si può o meno dedurre che $\Gamma \models \neg\phi$.*

Esercizio 140 *Considerare il seguente insieme Γ di formule:*

1. $\forall x(Dalmata(x) \rightarrow Cane(x))$
2. $\forall x(Cane(x) \rightarrow Mammifero(x))$
3 $\forall x(Mammifero(x) \rightarrow Animale(x))$

Γ rappresenta una gerarchia, cioè un albero la cui radice è etichettata da $Animale(x)$ e ciascun nodo del ramo, di cui $Dalmata(x)$ è foglia, è etichettato da un predicato. Estendere la gerarchia scrivendo opportune formule nella logica del primo ordine per aggiungere le foglie – e i nodi necessari per connettere le foglie alla radice etichettata da $Animale(x)$ – per i seguenti animali:

4. $Pitone(x)$,
5. $Balena(x)$,
6. $Delfino(x)$,
7. $Canguro(x)$.

[7]Quanto descritto nell'esercizio è un esempio usato comunemente nelle aree di ricerca dell'intelligenza artificiale che studiano il "ragionamento di senso comune".

(forse non è superfluo ricordare che balene e delfini sono mammiferi, mentre i canguri non lo sono).

Sia Γ^+ *l'insieme delle formule che rappresentano la gerarchia così estesa, unitamente alle seguenti formule:*

8. $Dalmata(pongo)$,
9. $Pitone(gustavo)$,
10. $Balena(margherita)$
11. $Delfino(giordano)$,
12 $Canguro(mario)$.

Verificare che la gerarchia ampliata sia tale che i seguenti sono soddisfatti:

a. $\Gamma^+ \models Mammifero(margherita)$
b. $\Gamma^+ \models \neg Mammifero(mario)$

Esercizio 141 *Definire una struttura* $\mathfrak{A} = \langle D, I \rangle$ *in cui la formula*

$$\forall x \exists y R(x, y)$$

sia vera.

Esercizio 142 *Definire una struttura* $\mathfrak{A} = \langle D, I \rangle$ *in cui la formula*

$$\forall x \forall y R(x, y)$$

sia vera.

Esercizio 143 *Dimostrare il lemma 6.21.*

Esercizio 144 *Completare la dimostrazione del teorema 6.23.*

Esercizio 145 *Dimostrare il teorema 6.24.*

Esercizio 146 *Dimostrare il teorema 6.25.*

Esercizio 147 *Dimostrare il teorema 6.26.*

Esercizio 148 *Dimostrare il teorema 6.27.*

6.3 Riepilogo

In questo capitolo abbiamo introdotto la *logica del primo ordine* o *calcolo dei predicati*. Abbiamo dapprima presentato:

1. il linguaggio, cioè l'insieme delle formule ben formate, del calcolo dei predicati. A tal fine abbiamo illustrato:
   (a) i simboli logici;
   (b) i simboli di costante, variabile, predicato e funzione;
   (c) la nozione di termine, di atomo e di formula;
   (d) i quantificatori esistenziale ed universale;
   (e) la nozione di variabile libera e legata;
   (f) abbiamo infine illustrato attraverso esempi l'uso dei quantificatori.
2. Abbiamo poi introdotto la semantica per il calcolo dei predicati, presentando:
   (a) la nozione di struttura di interpretazione;
   (b) la nozione di soddisfacibilità e conseguenza logica;
   (c) la nozione di modello;
   (d) abbiamo infine illustrato molte equivalenze semantiche in formule che coinvolgono quantificatori.

Capitolo 7
Sistemi deduttivi in logica del primo ordine

Nella logica proposizionale gli apparati deduttivi non sono essenziali per stabilire la validità di una formula, infatti per verificare se una formula proposizionale è una tautologia basta costruire la tabella dei valori di verità ad essa associata. Invece, come abbiamo notato alla fine del capitolo precedente, stabilire la validità di una formula nella logica del primo ordine è estremamente più complesso. La validità infatti comporta una verifica su *tutte* le possibili strutture, che possono essere in quantità non numerabile e con domini di qualunque cardinalità. Quindi, acquista particolare rilevanza la componente della logica che concerne l'apparato deduttivo e la sua correttezza e completezza rispetto alla semantica.

Avere a disposizione un apparato deduttivo corretto e completo rispetto alla semantica significa che, quando ci domandiamo se $\models A$, per rispondere possiamo utilizzare un metodo di dimostrazione, ovvero un metodo che utilizza una sequenza *finita* di argomenti che ci convincono che effettivamente A è valido. Oppure, quando ci domandiamo se $\Gamma \models A$, possiamo utilizzare un metodo di dimostrazione per verificare, con un numero finito di argomenti, che A consegue da Γ. E questo anche quando Γ è infinito, infatti una deduzione di una formula A da un insieme Γ è una sequenza finita $\langle A_1, \ldots, A_n \rangle$ tale che $A_n = A$, pertanto nella derivazione di A da Γ utilizziamo solo un insieme finito di ipotesi. Vedremo infatti che anche per la logica del primo ordine vale un teorema di compattezza. Inoltre, nella dimostrazione del teorema di completezza, vedremo che si può costruire un modello canonico per la logica del primo ordine, e che la verifica della validità di una formula si può di fatto ricondurre alla verifica su questo modello.

Un altro aspetto importante è la possibilità di enumerare effettivamente i teoremi, e perciò le formule valide. Presenteremo un teorema che dice che l'insieme delle formule valide è effettivamente enumerabile. Da notare che un ragionamento analogo non si può fare per i non teoremi, che non costituiscono un insieme ricorsi-

vamente enumerabile. Quindi l'insieme dei teoremi della logica del primo ordine è ricorsivamente enumerabile, ma non ricorsivo, cioè è semidecidibile.

Per potere trattare la nozione di deduzione nella logica del primo ordine dobbiamo approfondire due concetti. Il primo riguarda il modo in cui si comportano i termini, e le variabili libere che in essi compaiono, il secondo concerne il ruolo che hanno le tautologie proposizionali, nella deduzione di una formula del primo ordine da un insieme (eventualmente vuoto) di formule.

7.1 Sostituzioni

Introduciamo preliminarmente alcuni concetti. Come vedremo nel seguito, si possono sostituire termini al posto di variabili libere all'interno delle formule, tuttavia ciò è possibile sotto certe condizioni.

Definizione 7.1 *Una* sostituzione *è una funzione*[1] *dall'insieme delle variabili* VAR *all'insieme dei termini* TERM*, cioè* $\sigma : \text{VAR} \mapsto \text{TERM}$*. Dato un termine* t*,* $t\sigma$ *è definito ricorsivamente come segue:*

i. $\top\sigma = \top$ *e* $\bot\sigma = \bot$*;*

ii. se c *è un simbolo di costante,* $c\sigma = c$*;*

iii. se x *è un simbolo di variabile,* $x\sigma = \sigma(x)$*;*

iv. se f *è un simbolo di funzione di arità* n*, allora* $f(t_1, \ldots, t_n)\sigma = f(t_1\sigma, \ldots, t_n\sigma)$*.*

La sostituzione σ di un termine t al posto di un simbolo di variabile x è indicata da $\sigma = \{t/x\}$.

Lemma 7.2 *Se* t *è un termine e* σ *è una sostituzione, allora* $t\sigma$ *è un termine.*

Dimostrazione Per induzione strutturale su t.
(*Passo Base*) Se $t = c$ allora $t\sigma = c$ e quindi $t\sigma$ è un termine. Se $t = x$ allora $\sigma = \{t'/x\}$ o $\sigma = \{t'/y\}$ per qualche termine t', nel primo caso $x\sigma = \sigma(x) = t'$ e nel secondo $x\sigma = x$. In entrambi i casi $x\sigma$ è un termine.
(*Passo Induttivo*) Sia $t = f(s_1, \ldots, s_n)$, per definizione $t\sigma = f(s_1\sigma, \ldots, s_n\sigma)$; per ipotesi induttiva, per ogni i, $s_i\sigma$ è un termine t'_i , dunque $t\sigma = f(t'_1, \ldots, t'_n)$ è un termine.

□

Nel seguito considereremo solo quelle sostituzioni σ tale che $x\sigma \neq x$ e l'insieme di variabili che soddisfano tale condizione è finito. In altri termini parliamo di sostituzioni come insiemi $\{t_1/x_1, \ldots, t_n/x_n\}$, in cui, per ogni i, $x_i\sigma = t_i \neq x_i$.

Le sostituzioni si possono fare anche nelle formule, tenendo in conto il fatto che esse non possono modificare una variabile legata da un quantificatore. A questo fine indichiamo con σ_x una sostituzione che si comporta come σ tranne per il fatto che

[1] La scriveremo in notazione post-fissa.

non cambia la variabile x. Ovvero $\sigma_x(x) = x$ e $\sigma_x(y) = \sigma(y)$. Utilizzando questa notazione, possiamo estendere la nozione di sostituzione alle formule.

Definizione 7.3 *Data una formula ϕ la formula $\phi\sigma$ è definita nel seguente modo:*

i. *se $\phi = P(t_1, \ldots, t_n)$ è una formula atomica, allora $\phi\sigma = P(t_1, \ldots, t_n)\sigma = P(t_1\sigma, \ldots, t_n\sigma)$;*

ii. *se $\phi = (\neg A)$ allora $\phi\sigma = (\neg A)\sigma = \neg(A\sigma)$;*

iii. *se $\phi = (A \circ B)$ allora $\phi\sigma = (A \circ B)\sigma = A\sigma \circ B\sigma$, per ogni connettivo binario $\circ$;*

iv. *se $\phi = (\forall x A)$ allora $\phi\sigma = (\forall x A)\sigma = \forall x(A\sigma_x)$;*

v. *se $\phi = (\exists x A)$ allora $\phi\sigma = (\exists x A)\sigma = \exists x(A\sigma_x)$.*

Nel seguito useremo la convenzione che se $\sigma = \{t/x\}$ allora scriveremo $A[t/x]$ invece di $A\sigma$.
Consideriamo il seguente esempio.

Esempio 64 *Sia $\sigma = \{a/x, b/y\}$. Allora*

$$\begin{aligned}[\forall x R(x,y) \to \exists y R(x,y)]\sigma &= [\forall x R(x,y)]\sigma_x \to [\exists y R(x,y)]\sigma_y \\ &= [\forall x R(x,b)] \to [\exists y R(a,y)]\end{aligned} .$$

Introduciamo ora la nozione di *termine libero per una variabile.*

Definizione 7.4 *Se A è una formula, t un termine e $y_1, \ldots, y_n$ le variabili di t, allora si dice che t è* libero *per la variabile x nella formula A se nessuna occorrenza libera di x in A si trova nel campo d'azione di un quantificatore $\forall y_i$.*

Esempio 65 *Consideriamo il termine $t = f(x,y)$ e la formula $A = \forall x(P(x) \to Q(y))$; la variabile y occorre libera in A, ma il termine $f(x,y)$ non è libero per y perché la x del termine t si trova nel campo di azione del quantificatore.*

Il termine x è libero per y nella formula $A = Q(y)$, ma non lo è nella formula $A = \forall x P(x,y)$; infatti, se sostituiamo la y con la x in A otteniamo la formula $\forall x P(x,x)$ di molto differente dall'originale.

Se σ_1 e σ_2 sono due sostituzioni, essendo delle funzioni, possiamo parlare di composizione: essa è denotata da $\sigma_1 \circ \sigma_2$ o anche $\sigma_1\sigma_2$. Si può dimostrare (si veda l'esercizio 152) che la composizione di sostituzioni è associativa, ovvero se σ e τ sono sostituzioni, si ha che per ciascun termine t $(t\sigma)\tau = t(\tau\sigma)$. Le nozioni appena introdotte sono necessarie per garantire l'associatività delle sostituzioni. Senza considerare la nozione di *termine libero* per una formula e quindi di *sostituzione libera* per una formula, l'associatività non è verificata.

Esempio 66 *Consideriamo la formula* $\forall y R(x,y)$ *e le sostituzioni* $\sigma = \{y/x\}$ *e* $\tau = \{c/y\}$. *Pertanto* $\sigma\tau = \{c/x, c/y\}$. *Ora, se facciamo sostituzioni non libere abbiamo:* $\forall y R(x,y)\sigma_y = \forall y R(y,y)$ *e* $\forall y R(y,y)\tau_y = \forall y R(y,y)$ *o* $(\forall y R(x,y)\sigma_y)\tau_y = \forall y R(y,y)$.

Ma $\forall y R(x,y)(\sigma_y\tau_y) = \forall y R(x,y)(\{c/x, c/y\})_y = \forall y R(c,y)$. *Quindi non varrebbe l'associatività delle sostituzioni.*

Definizione 7.5 *La nozione di* sostituzione σ, libera per una formula A, *è definita induttivamente come segue:*

1. *Se* A *è un atomo,* σ *è libera per* A;
2. σ *è libera per* $(\neg A)$ *se* σ *è libera per* A;
3. σ *è libera per* $(A \circ B)$ *se* σ *è libera per* A *e per* B;
4. σ *è libera per* $\forall x A$ $(\exists x A)$ *se* σ_x *è libera per* A *e se, per ogni variabile* y *diversa da* x, $y\sigma$ *è un termine libero per* x *in* A.

Esempio 67 *Nell'esempio 66 il termine* $x\sigma$ *non è libero per la variabile* y.

7.1.1 Esercizi

Esercizio 149 *Dire se il termine* $y + z$ *è libero per* x *in*

1. $\forall xz(P(x,z) \wedge Q(x,z))$
2. $\forall tw(P(t,w) \wedge Q(x,z))$
3. $\forall yz(P(y,z) \wedge Q(x,z))$.

Esercizio 150 *Individuare le sostituzioni libere che permettono di trasformare le seguenti formule:* a_1 *in* a_2, b_1 *in* b_2 *e* c_1 *in* c_2:

a_1 $\forall x(P(x) \wedge Q(y,z) \wedge R(f(z))$

a_2 $\forall x(P(x) \wedge Q(a,b) \wedge R(f(b))$

b_1 $\forall x P(x,y,z)$

b_2 $\forall x P(x,z,y)$

c_1 $W(x,y,x)$

c_2 $W(a+b, x, a+b)$.

Esercizio 151 *Dire se le formule 2, 3, 4 e 5 possono essere ottenute per sostituzione dalla 1, e in caso positivo individuare le sostituzioni:*

1. $P(x, y, x)$
2. $P(a, b, a)$
3. $P(a, b, b)$
4. $P(a, a, a)$
5. $P(a + b, c, b + a)$.

Esercizio 152 *Dimostrare che, se σ è una sostituzione libera per la formula ϕ e τ è una sostituzione libera per $\phi\sigma$ allora $(\phi\sigma)\tau = \phi(\sigma\tau)$.*

Esercizio 153 *Provare che, se σ è una sostituzione libera per la formula ϕ:*

1. $\phi \models \phi\sigma$
2. $\forall x\phi \models \forall x\phi\sigma$

Esercizio 154 *Data la seguente formula F:*

$$F = \neg\forall y(x = y) \tag{7.1}$$

Scrivere per esteso le formule

$$\forall xF \to F[z/x]$$
$$\forall xF \to F[y/x]$$

7.2 Tautologie e sostituzione uniforme

Come nel calcolo proposizionale, anche nella logica del primo ordine utilizziamo la regola di *sostituzione uniforme* (SU).

Definizione 7.6 *Data una formula proposizionale A, una formula A' di un linguaggio del primo ordine $\mathcal{L}$, si dice* istanza *di A se A' è il risultato della sostituzione uniforme di formule di $\mathcal{L}$ al posto dei simboli proposizionali in A.*

Si può facilmente dimostrare che la logica del primo ordine *estende* la logica proposizionale, nel senso che le tautologie proposizionali sono schemi validi anche nella logica del primo ordine, cioè le formule ottenute per sostituzione uniforme da una tautologia del calcolo proposizionale sono valide. L'istanza di una tautologia proposizionale è una formula valida del primo ordine, ma, come già osservato nel paragrafo 6.2.1 non è vero il viceversa, cioè non tutte le formule valide del primo ordine sono istanze di tautologie proposizionali.

Abbiamo visto, nel calcolo proposizionale, che per determinare la validità di una formula è sufficiente considerare solo un numero finito di assegnazioni di valori di verità alle lettere proposizionali. Per ciascuna di tali assegnazioni l'estensione alle formule può essere fatta seguendo le tabelle di verità. Abbiamo visto che, per questo, la logica proposizionale è decidibile.

Differentemente da una tale procedura finitaria, nella logica del primo ordine dobbiamo considerare non solo tutte le strutture $\mathfrak{A}$ del linguaggio (ovvero ogni insieme non vuoto), ma per ciascuna di tali strutture dobbiamo prendere in considerazione tutte le possibili assegnazioni η alle varibili che soddisfano, in $\mathfrak{A}$, la formula A in questione. Quindi se il dominio D è infinito tutto ciò non è effettivamente realizzabile. In altri termini, l'insieme delle formule valide della logica del primo ordine non è decidibile. Tuttavia la nozione di validità, come vedremo nel prossimo paragrafo, è equivalente alla nozione di derivabilità, per mezzo della quale saremo in grado di mostrare che l'insieme delle formule valide è effettivamente enumerabile.

Si può quindi apprezzare come la nozione sintattica di derivabilità e la dimostrazione di correttezza e completezza di un apparato deduttivo acquistino nella logica del primo ordine una importanza ben maggiore di quanta ne avessero nel calcolo proposizionale.

7.2.1 Esercizi

Esercizio 155 *Dimostrare che la formula del primo ordine*

$$(\forall x \exists y P(x,y)) \rightarrow ((\exists y \forall x P(x,y)) \rightarrow (\forall x \exists y P(x,y)))$$

ottenuta dalla tautologia proposizionale $P \rightarrow (Q \rightarrow P)$ *per mezzo di sostituzione uniforme è una formula valida.*

Esercizio 156 *Fornire tre esempi di formule valide del primo ordine che non siano istanze di tautologie proposizionali.*

7.3 Sistema hilbertiano in logica del primo ordine

In questo paragrafo introduciamo l'apparato deduttivo di Hilbert per la logica del primo ordine.

Nello stabilire l'insieme dei teoremi della logica del primo ordine, si parte da un insieme costituito dai tre *schemi di assioma* del calcolo proposizionale (si veda il paragrafo 4.1) cui ne vengono aggiunti due che trattano le formule quantificate.

Definizione 7.7 *L'insieme* AX-FOL *degli* assiomi logici *del sistema hilbertiano è l'insieme di tutte le formule che si ottengono sostituendo uniformemente formule al posto delle variabili A B e C nei seguenti schemi:*

1. $(A \to (B \to A))$;
2. $((A \to (B \to C)) \to ((A \to B) \to (A \to C)))$;
3. $((B \to \neg A) \to ((B \to A) \to \neg B))$;
4. $\forall x A[x] \to A[t/x]$ *dove* t *è libero per* x *in* A.
5. $\forall x(A \to B) \to (A \to \forall x B)$ *se* A *è una formula che non contiene occorrenze libere di* x.

L'insieme delle regole di inferenza *del sistema hilbertiano è costituito da:*

modus ponens (MP)*:*

$$\frac{A \quad A \to B}{B}$$

generalizzazione (Gen)*:*

$$\frac{A}{\forall x A}$$

L'insieme degli *assiomi logici* e delle *regole di inferenza* sopra definiti costituisce un sistema formale assiomatico, chiamato *sistema di Hilbert.*

Dunque una struttura per il sistema assiomatico di Hilbert è una struttura che soddisfa gli assiomi sopra definiti. Tale struttura esiste sempre, ovvero un calcolo dei predicati con gli assiomi sopra introdotti è soddisfacibile. Per costruire una struttura siffatta è sufficiente prendere $\mathfrak{A} = \langle D, I \rangle$ con D costituito da un solo elemento.

Dimostriamo ora, come abbiamo fatto per il calcolo proposizionale, che i teoremi del sistema di Hilbert sono esattamente le formule logicamente valide; ovvero che la chiusura deduttiva di AX-FOL secondo MP e Gen coincide con l'insieme delle formule valide. Cioè dimostriamo che l'apparato deduttivo è corretto e completo.

La nozione di *dimostrazione* si può facilmente mutuare da quella fornita per il calcolo proposizionale. Ugualmente tutte le nozioni e proprietà date a riguardo dell'operatore $\vdash_H$ nei capitoli precedenti si possono estendere alla logica del primo ordine. In particolare la nozione di consistenza.

Tuttavia, differentemente dal caso proposizionale, il teorema di deduzione non vale nella sua forma generale, come si può vedere dal prossimo esempio:

Esempio 68 *Consideriamo l'atomo $P(x)$, abbiamo $P(x) \vdash P(x)$; per Gen abbiamo $P(x) \vdash \forall x P(x)$, tuttavia $\not\vdash P(x) \to \forall x P(x)$. Prendiamo un modello $\mathfrak{A} = \langle \{2,3,4\}, \{2,4\} \rangle$, e una assegnazione η, tale che $x^\eta = 4$. Abbiamo $\mathfrak{A}, \eta \models P(x)$ ma non è vero che $\mathfrak{A}, \eta \models \forall x P(x)$*

Pertanto è proprio la regola di generalizzazione che rende il teorema di deduzione inapplicabile nella sua formulazione generale, ma esso vale sotto certe restrizioni. A tal fine dobbiamo introdurre la nozione di *dipendenza* di una formula da altre.

Definizione 7.8 *Siano* Γ *un insieme di formule e* A *una formula di* Γ*, sia* $S = \langle A_1, \ldots, A_n \rangle$ *una sequenza di formule in* Γ*, diciamo che* A_i dipende da A nella sequenza S *se:*

i. $A_i = A \in \Gamma$*;*

ii. A_i *è ottenuto per MP o Gen da precedenti formule in* S *dove alcune di tali formule dipendono da* A*.*

In sostanza la nozione di dipendenza di una formula da un altra è un modo per costruire un legame tra due formule in una dimostrazione e serve per circoscrivere l'applicazione della regola Gen:

Teorema 7.9 [TEOREMA DI DEDUZIONE] *Si assuma che* $\Gamma, A \vdash B$ *e che, nella deduzione di* B *da* $\Gamma \cup \{A\}$*, l'applicazione della regola Gen a una formula* B_i *che dipende da* A *coinvolga solo variabili che non occorrono libere in* A *e quantificate in* B_i*, allora*

$$\Gamma \vdash A \rightarrow B.$$

Dimostrazione La dimostrazione è analoga al caso proposizionale, dunque ci limitiamo a dimostrare il caso in cui qualche B_i è ottenuto per generalizzazione da qualche B_j, $j < i$, con $B_i = \forall x B_j$, nella sequenza $B_1 \ldots, B_n = B$ di formule che costituisce la dimostrazione di B da $\Gamma \cup \{A\}$.

Per ipotesi induttiva $\Gamma \vdash A \rightarrow B_j$ e, per ipotesi del teorema, B_j non dipende da A oppure x non è una variabile libera di A.

Se B_j non dipende da A allora, per induzione sulla dimostrazione di B_j da Γ e per definizione induttiva di dipendenza, abbiamo $\Gamma \vdash B_j$ e quindi, per generalizzazione, abbiamo $\Gamma \vdash \forall x B_j$. Pertanto $\Gamma \vdash B_i$.

Se B_j dipende da A, allora per ipotesi x non è una variabile libera di A; allora per lo schema d'assioma e (cioè $\vdash \forall x(A \rightarrow B_j) \rightarrow (A \rightarrow \forall x B_j)$) e per il fatto che da $\Gamma \vdash A \rightarrow B_j$, per Gen, si ha che: $\Gamma \vdash \forall x(A \rightarrow B_j)$. Mediante MP otteniamo che $\Gamma \vdash (A \rightarrow \forall x B_j)$, ovvero che $\Gamma \vdash (A \rightarrow B_i)$. □

È evidente che se A è una formula chiusa le ipotesi del teorema 7.9 sono automaticamente soddisfatte, abbiamo quindi il seguente corollario:

Corollario 7.10 *Se* A *è un enunciato allora*

$$\Gamma, A \vdash B \text{ sse } \Gamma \vdash A \rightarrow B.$$

Definizione 7.11 *Un insieme di enunciati* Γ *in* $\mathcal{L}$ è consistente *se non esiste una derivazione di* A *e* $\neg A$ *da* Γ*; cioè se* $\Gamma \nvdash A \wedge \neg A$*.*

La nozione di consistenza per un insieme di enunciati del primo ordine è analoga a quella fornita nel caso proposizionale.

Prima di presentare il teorema di correttezza introduciamo un lemma che ci servirà nella sua dimostrazione.

Lemma 7.12

1. *Ogni assioma logico è valido;*
2. *le regole di inferenza conservano la validità.*

Dimostrazione Lasciata al lettore (si veda l'esercizio 157). □

Teorema 7.13 [CORRETTEZZA] *Se* $\Gamma \vdash_H A$ *allora* $\Gamma \models A$.

Dimostrazione Usando il lemma 7.12, si dimostra per induzione che ogni formula A che si può dedurre da Γ è logicamente implicata da Γ. Abbiamo i seguenti casi:

1. A è un assioma logico e quindi per il lemma 7.12 $\models A$ e a maggior ragione $\Gamma \models A$;
2. $A \in \Gamma$ e quindi ovviamente $\Gamma \models A$;
3. $A = \forall x B$ è ottenuto per Gen. Per ipotesi induttiva $\Gamma \models B$ e per il lemma 7.12 $\Gamma \models \forall x B$; quindi $\Gamma \models A$;
4. A è ottenuta per MP da A_k e $A_k \rightarrow A$. Per ipotesi induttiva abbiamo che $\Gamma \models A_k$ e $\Gamma \models \mathcal{A}_k \rightarrow A$. Di nuovo per il lemma 7.12, $\Gamma \models A$.

□

Teorema 7.14 [COMPLETEZZA, GÖDEL 1930]

a) Se $\Gamma \models A$ *allora* $\Gamma \vdash_H A$*;*

b) Ogni insieme consistente di formule ha un modello.

Dimostrazione La dimostrazione, come nel caso del calcolo proposizionale, si basa sulla costruzione di un contromodello per A ovvero di un modello per $\Gamma \cup \{\neg A\}$. Pertanto, con l'argomentazione che segue, dimostriamo anche che i punti a e b sono equivalenti.

La dimostrazione viene fatta con i seguenti passi:

1. Sia $\mathcal{L}$ il linguaggio di Γ che supponiamo numerabile. Per ipotesi $\Gamma \cup \{\neg A\}$ è consistente, perché $\Gamma \nvdash A$. Estendiamo il linguaggio $\mathcal{L}$ al linguaggio $\mathcal{L}(C)$ che contiene un insieme numerabile di nuovi simboli di costante c_i.
2. Per ogni formula esistenziale di $\mathcal{L}$, utilizzando le costanti si aggiungono dei *testimoni* per le variabili esistenziali nel seguente modo:

$$\begin{aligned} \Delta_0 &= \Gamma \cup \{\neg A\} \\ \Delta_i &= \{\exists x \phi(x) \rightarrow \phi(c_i)\} \cup \Delta_{i-1} \\ &\quad \text{per ciascuna formula } \exists x \phi(x) \in \mathcal{L}, \\ &\quad \text{dove } c_i \text{ è un simbolo di costante che non compare in } \Delta_{i-1}. \end{aligned}$$

3. Si mostra che ogni Δ_i è consistente, in quanto supponendo Δ_{i-1} consistente, se $\Delta_{i-1} \cup \{\exists x\phi(x) \rightarrow \phi(c_i)\}$ è inconsistente, segue che $\Delta_{i-1} \vdash \neg(\exists x\phi(x) \rightarrow \phi(c_i))$ e, siccome c_i non compare in Δ_{i-1} allora $\Delta_{i-1} \vdash \forall y\neg(\exists x\phi(x) \rightarrow \phi(y))$ ovvero Δ_{i-1} è inconsistente; contraddizione.
4. Si mostra che l'insieme $\Delta = \cup\Delta_i$ è consistente.
5. Δ può quindi essere esteso a un insieme consistente e massimale Δ' che, pertanto estende $\Gamma \cup \{\neg A\}$. Osserviamo che Δ' è un insieme di enunciati che contengono testimoni.
6. Si mostra che Δ ha un modello particolare, che chiameremo modello *canonico*, costruito su classi di equivalenza di enunciati, questo passo è necessario per potere interpretare l'uguaglianza. Infatti si costruisce il modello canonico $\mathfrak{A}$ prendendo come dominio le classi di equivalenza dei termini di $\mathcal{L}(C)$.
7. Il modello canonico risulterà essere un modello di $\Gamma \cup \{\neg A\}$.
8. A questo punto abbiamo mostrato che $\Gamma \not\models A$ e quindi la completezza.

□

Teorema 7.15 [COMPATTEZZA]

i. Se $\Gamma \models A$ *allora per qualche insieme finito* $\Gamma_0 \subseteq \Gamma$, $\Gamma_0 \models A$;

ii. Se ogni sottoinsieme finito Γ_0 *di* Γ *è soddisfacibile allora* Γ *è soddisfacibile.*

Dimostrazione

i. Si consideri che:

$$\begin{array}{lll} \Gamma \models A & \text{implica } \Gamma \vdash A & \text{per il teorema 7.14 (completezza)} \\ & \text{implica } \Gamma_0 \vdash A & \text{per qualche sequenza finita } \Gamma_0 = A_1, \ldots, A_n \\ & \text{implica } \Gamma_0 \models A & \text{per il teorema 7.13 (correttezza).} \end{array}$$

ii. Se ogni sottoinsieme finito Γ_0 di Γ è soddisfacibile allora, per la correttezza, ogni sottoinsieme finito Γ_0 è consistente. Se per ogni Γ_0 esiste un A tale che $\Gamma_0 \not\vdash A$ allora Γ è consistente. Per la nozione finitaria di dimostrazione e per la completezza, Γ è soddisfacibile.

□

Per dimostrare la completezza, nella costruzione dell'insieme Δ, si usa una tecnica analoga a quella usata per la compattezza nel caso proposizionale. L'unica differenza sta nel fatto che l'insieme massimale viene costruito usando un insieme di testimoni.

Per la logica dei predicati si può dimostrare che la relazione di conseguenza logica è ricorsivamente enumerabile, sotto determinate condizioni.

Teorema 7.16 *L'insieme delle dimostrazioni di una formula valida di un linguaggio numerabile del primo ordine è decidibile.*

Dimostrazione Sia $\mathbb{D}$ l'insieme delle dimostrazioni di A. Proviamo che esso è decidibile.

Innanzi tutto ricordiamo che, essendo il linguaggio $\mathcal{L}$ numerabile ed essendo FBF ricorsivo, l'insieme delle formule di $\mathcal{L}$ è decidibile. Quindi, data una qualunque sequenza finita $S = \langle A_1, \ldots, A_n \rangle$, per ogni A_i possiamo decidere se essa è una formula o no.

Se, per qualche i A_i non è una formula, allora $S \notin \mathbb{D}$. Altrimenti, se tutti gli elementi di S sono formule, per stabilire se S è o no una dimostrazione di A dobbiamo stabilire se $A_n = A$ e se A_i è ottenuta per mezzo di regole di inferenza da precedenti A_j, $j < i$ o come istanza di schema di assioma. Se S non passa qualcuna di queste verifiche (ciascuna delle quali è decidibile) allora $S \notin \mathbb{D}$. Altrimenti, $S \in \mathbb{D}$.

Pertanto $\mathbb{D}$ è decidibile. □

Teorema 7.17 *L'insieme delle formule valide di un linguaggio numerabile del primo ordine può essere effettivamente enumerato.*

Dimostrazione Per il teorema precedente possiamo enumerare tutte le sequenze di formule che costituiscono dimostrazioni di $\mathcal{L}$. Ciascuna di esse è della forma $S_i = \langle A_1, \ldots, A_{n_i} \rangle$ e si conclude con il teorema $A = A_{n_i}$. Questo ci permette di enumerare effettivamente i teoremi.

Siccome i teoremi sono tutte e sole le formule valide, abbiamo enumerato le formule valide. □

Quanto abbiamo visto nell'ipotesi che una formula sia valida, cioè conseguenza logica di un Γ vuoto, può essere esteso al caso in cui Γ è un insieme decidibile:

Corollario 7.18 *Se Γ è un insieme decidibile in un linguaggio enumerabile, allora l'insieme delle conseguenze logiche di Γ è effettivamente enumerabile.*

□

Quindi, ovviamente, l'insieme dei teoremi è effettivamente enumerabile.

Inoltre, se Γ è decidibile ed è una teoria completa , ovvero $\Gamma \vdash A$ oppure $\Gamma \vdash \neg A$ per ogni A (si veda la definizione 4.5), allora l'insieme delle formule logicamente implicate da Γ è decidibile.

D'altro canto abbiamo anche il seguente teorema dovuto a Church:

Teorema 7.19 [INDECIDIBILITÀ DELLA LOGICA DEI PREDICATI, CHURCH] *Sia $\mathcal{L}$ un linguaggio numerabile con almeno un simbolo di costante c, e un simbolo di predicato binario P. L'insieme delle formule valide non è decidibile.*

La dimostrazione di questo teorema originariamente è stata fatta sul linguaggio della teoria elementare dei numeri (si veda esempio 47), tuttavia si è mostrato che è sufficiente che il linguaggio contenga un simbolo di predicato binario. D'altro canto se $\mathcal{L}$ ha solo $\forall$ e un simbolo di predicato unario, allora l'insieme delle formule valide è decidibile.

7.3.1 Esercizi

Esercizio 157 *Dimostrare il lemma 7.12.*

Esercizio 158 *Verificare che dal teorema 7.15 si deriva che se* Γ *ha un modello allora ogni sottoinsieme finito di* Γ *ha un modello. In particolare, verificare che i punti i e ii del teorema 7.15 sono equivalenti.*

Esercizio 159 *Sia* $\Sigma = \{\phi | \mathfrak{A} \models \phi$ *e* $\mathfrak{A}$ *è finito*$\}$*, mostrare che l'insieme* Σ *è effettivamente enumerabile.*

Esercizio 160 *Sia* $\mathcal{L}$ *un linguaggio con un numero finito di parametri. Sia* Σ *un insieme di enunciati tale che, per ogni* $\phi \in \Sigma$ *se* $\not\models \phi$ *allora esiste un modello finito* $\mathfrak{A}$ *tale che* $\mathfrak{A} \models \neg\phi$*. Mostrare che* Σ *è decidibile esibendo una procedura effettiva che, per ogni* $\phi \in \Sigma$*, stabilisca se* ϕ *è valido o meno.*

Esercizio 161 *Sia* Φ *l'insieme delle formule* $\phi = \forall x_1 \cdots \forall x_n \psi$ *dove* ψ *non contiene né simboli di funzione né quantificatori. Mostrare* Φ *è decidibile.*

7.4 Deduzione naturale in logica del primo ordine

Il sistema deduttivo della deduzione naturale si estende alla logica del primo ordine aggiungendo alle regole per trattare le formule proposizionali presentate in 4.3, nuove regole per trattare le formule quantificate universalmente ed esistenzialmente.

Per le formule quantificate universalmente vengono introdotte le due regole seguenti:

$$(\forall e) \quad \frac{\forall x A}{A[t/x]}$$

$$(\forall i) \quad \frac{A[y/x]}{\forall x A}$$

dove x, y sono variabili e t è un generico termine. La regola di eliminazione del quantificatore universale ha un'ovvia lettura. Quella per l'introduzione è una regola condizionale: essa si può applicare purché la variabile y non compaia libera in nessuna delle formule che compaiono nelle foglie non cancellate del sottoalbero la cui radice è $A[y/x]$. Questo, informalmente, significa che y non è stata usata in modo essenziale nella derivazione di $A[y/x]$ e non sono state fatte su di essa ipotesi particolari, altrimenti non sarebbe più "libera", quindi non sarebbe lecito chiuderla universalmente.

Per le formule quantificate esistenzialmente vengono introdotte le due regole seguenti:

$$(\exists e)\ \frac{\exists x A \qquad \dfrac{[A[y/x]]}{C}}{C}$$

$$(\exists i)\ \frac{A[t/x]}{\exists x A}$$

In $(\exists e)$ la variabile y non può comparire in C, né in nessuna delle foglie non cancellate del sottoalbero di radice C, a parte $A[y/x]$. La restrizione ha un carattere analogo a quella precedentemente fatta per la regola $(\forall i)$, cioè nella dimostrazione di C, su y non debbono essere state fatte ipotesi particolari.

Mostriamo ora un esempio di derivazione:

Esempio 69 *Dimostrare con la deduzione naturale che:*

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

$$\dfrac{\dfrac{\dfrac{[\forall x P(x)]}{P(y)}\ (\forall e)}{\exists y P(y)}\ (\exists i)}{\forall x P(x) \to \exists x P(x)}\ (\to i)$$

Il lettore può facilmente verificare che le condizioni per l'applicabilità delle regole $(\forall i)$ ed $(\exists e)$ sono essenziali per la correttezza dell'apparato deduttivo della deduzione naturale; infatti rilassando le condizioni per l'applicabilità di dette regole si possono derivare non teoremi, quali ad esempio: $\exists x P(x) \to \forall x P(x)$ (si veda l'esercizio 163).

Per la deduzione naturale si possono dimostrare correttezza e completezza; come nel caso proposizionale le dimostrazioni possono essere fatte direttamente o riconducendo la deducibilità in deduzione naturale a quella nel sistema hilbertiano.

7.4.1 Esercizi

Esercizio 162 *Dimostrare, usando la deduzione naturale che:* $\vdash \forall x(P(x) \to Q) \to (\exists x P(x) \to Q)$.

Esercizio 163 *Dimostrare che, rilassando le condizioni per l'applicabilità delle regole* $(\forall i)$ *ed* $(\exists e)$ *può derivare* $\exists x P(x) \to \forall x P(x)$.

7.5 Riepilogo

In questo capitolo, analogamente a quanto fatto nel capitolo 4 per la logica proposizionale, abbiamo presentato il *sistema hilbertiano* quale apparato deduttivo per la logica del primo ordine.

1. Per questa abbiamo dato il teorema di deduzione, di correttezza e completezza della deduzione rispetto alla conseguenza logica ($\vdash_H \equiv \models$);
2. abbiamo enunciato il teorema di compattezza;
3. abbiamo mostrato che l'insieme delle formule valide della logica del primo ordine è ricorsivamente enumerabile;
4. abbiamo infine mostrato che il calcolo dei predicati non è decidibile, ma solo semidecidibile.

Abbiamo poi presentato le regole della *deduzione naturale* per il primo ordine, con esempi di deduzioni.