

Università di Bologna
Dipartimento di Informatica

Appunti di:

Logica per l'informatica

Andrea Manzo

A.S. 2025 / 2026

Contents

1	Formule logiche	4
1.1	Proposizioni	4
1.2	Introduzione alla logica del prim'ordine	4
1.2.1	Sintassi	4
1.2.2	Connettivi e Quantificatori	5
1.2.3	Precedenza e Associatività dei Connettivi	5
2	Teoria degli insiemi	6
2.1	Teoria degli insiemi naive	6
2.2	Teorie assiomatiche degli insiemi	6
2.3	Teoria di Zermelo-Fraenkel (ZF)	7
2.3.1	Assioma di Estensionalità	7
2.3.2	Definizione di Sottoinsieme	7
2.3.3	Assioma di separazione	7
2.3.4	Assioma dell'insieme vuoto	8
2.3.5	Definizione dell'insieme vuoto	8
2.3.6	Definizione e teorema di intersezione binaria	8
2.3.7	Definizione di intersezione	9
2.3.8	Assioma dell'unione	9
2.3.9	Unione binaria (teorema)	10
2.3.10	Assioma del singoletto	10
2.3.11	Costruzione dei numeri naturali	11
2.3.12	Assioma dell'infinito	12
2.3.13	Teorema dell'esistenza di \mathbb{N}	12
2.3.14	Assioma dell'insieme potenza	12
2.3.15	Assioma di regolarità (o di fondazione)	12
2.3.16	Assioma di rimpiazzamento	12
3	Regole di dimostrazione	13
3.1	\forall -Introduzione	13
3.2	\forall -Eliminazione	13
3.3	\Rightarrow -Introduzione	13
3.4	\Rightarrow -Eliminazione	14
3.5	\Rightarrow -Eliminazione (variante)	14
3.6	\Leftrightarrow -Introduzione	14
3.7	\Leftrightarrow -Eliminazione	14
3.8	Abbreviazioni	15
3.9	Espansione di definizioni	15
3.10	Esplicitazione della conclusione	15
3.11	Ovvio	16
3.12	Assurdo-Eliminazione	16
3.13	\neg -Introduzione	16
3.14	\neg -Eliminazione	16
3.15	\wedge -Introduzione	17
3.16	\wedge -Eliminazione	17
3.17	\vee -Introduzione	17
3.18	\vee -Eliminazione	18
3.19	Risultati intermedi	18
3.20	\exists -Introduzione	18
3.21	\exists -Eliminazione	19
4	Teoremi e Dimostrazioni	20
4.1	Teorema di Riflessività del \subseteq	20
4.2	Teorema di Anti-simmetria del \subseteq	20
4.3	Teorema di Transitività del \subseteq	21
4.4	Teorema del Vuoto come sottoinsieme di ogni cosa	22
4.5	Teorema del vuoto come unico sottoinsieme del vuoto	22

4.6	Teorema dell'intersezione con il vuoto	23
4.7	Teorema della Monotonia dell'intersezione	24
4.8	Teorema della monotonia dell'unione	24
5	Elementi matematici, relazioni, funzioni	26
5.1	Copie ordinate	26
5.1.1	Teorema di caratterizzazione delle coppie	26
5.1.2	Corollario del teorema delle coppie	26
5.2	Prodotto cartesiano di insiemi	26
5.3	Relazioni	26
5.3.1	Relazioni da e verso insiemi vuoti	27
5.4	Funzioni	27
5.4.1	Spazio di funzioni	27
5.5	Proprietà delle relazioni	27
5.6	Tipi di relazioni	28
5.7	Classi di equivalenza	29
5.8	Insieme quoziente	29
5.8.1	Esempio: i numeri interi \mathbb{Z}	30
5.9	Proprietà delle funzioni	30
5.10	Cardinalità	31
5.10.1	Numeri cardinali	31
5.10.2	Insiemi enumerabili	31
5.10.3	Definizione di cardinalità	32
5.10.4	Ordinamento dei numeri cardinali	32
5.11	Insiemi finiti e infiniti	32
5.12	Teorema di Cantor	32
6	Semantica classica	33
6.1	Terminologia	33
6.2	Semantica classica della logica proposizionale	33
6.3	Logica classica	33
6.4	Conseguenza logica	34
7	Semantica intuizionista	36
7.1	Introduzione	36
7.2	Semantiche	36
7.3	Decidibilità, correttezza e completezza	37
8	Deduzione naturale	38
8.1	Logica proposizionale	38
8.2	Alberi di deduzione naturale	39
8.3	Sintassi per le regole di inferenza	40
8.4	Correttezza delle regole di inferenza	40
8.5	Invertibilità delle regole	41
8.6	Derivabilità delle regole	41
8.7	Regole di dimostrazione	41
8.7.1	Introduzione di \wedge	41
8.7.2	Eliminazione di \wedge	42
8.7.3	Regole alternative di eliminazione di \wedge	42
8.7.4	Introduzione di \vee	42
8.7.5	Eliminazione di \vee	43
8.7.6	Introduzione del \perp	43
8.7.7	Eliminazione del \perp	43
8.7.8	Introduzione del \top	43
8.7.9	Eliminazione del \top	44
8.7.10	Introduzione di \Rightarrow	44
8.7.11	Eliminazione di \Rightarrow	44
8.7.12	Introduzione del \neg	44
8.7.13	Eliminazione del \neg	45

8.8	Correttezza e completezza delle regole	45
8.8.1	Regola di dimostrazione RAA	45
8.8.2	Dimostrazione tramite EM	46
9	Logica del prim'ordine	47
9.1	Sintassi	47
9.2	Semantica classica della logica del prim'ordine	47
9.3	Binder	47
9.4	α -convertibilità	48
9.5	Sostituzione	49
10	Deduzione naturale per la logica del prim'ordine	50
10.1	Introduzione del \forall	50
10.2	Eliminazione del \forall	50
10.3	Introduzione del \exists	50
10.4	Eliminazione del \exists	51
11	Ricorsione e induzione	52
11.1	Introduzione	52
11.2	Tipi di dato	52
11.3	Programmi	53
11.4	Pattern matching	54
11.5	Ricorsione strutturale	54
11.6	Induzione strutturale	55

1 Formule logiche

1.1 Proposizioni

Una **proposizione** è una frase alla quale ha senso chiedersi se sia vera o falsa, se valga o meno.

Esempio

“2 è un numero pari.” è una proposizione.

Fraasi come “mangia la mela!” o “2” non sono proposizioni, poiché non si può stabilire un valore di verità per esse.

In ambito logico, esistono **linguaggi formali** che hanno regole rigide per la scrittura delle proposizioni, e queste proposizioni vengono poi usate nelle dimostrazioni per formulare **teoremi**. Un teorema è una proposizione che è stata dimostrata attraverso un ragionamento corretto.

Nel corso di una dimostrazione, l'enunciato che stiamo cercando di provare può evolversi: partiamo da ipotesi che vengono assunte come vere per restringere il campo di applicazione e arriviamo alla conclusione che dipende dalle ipotesi fatte. Questo tipo di ragionamento, definito “**ipotetico**”, implica che, nel momento in cui facciamo un'ipotesi, non stiamo affermando che essa sia **vera in senso assoluto**, ma solo che la consideriamo valida all'interno di un certo insieme di situazioni: le ipotesi riducono i casi possibili, permettendo di concentrarsi su situazioni specifiche per trarre conclusioni.

1.2 Introduzione alla logica del prim'ordine

In generale esistono varie logiche, la **logica del prim'ordine** è una delle più semplici e una delle più utilizzate.

Ora, nella spiegazione della sintassi della logica del prim'ordine useremo una spiegazione Tarskiana, ovvero per spiegare un connettivo useremo lo stesso connettivo del meta-livello, senza aggiungere niente di nuovo. Questa spiegazione presuppone che le due logiche siano coerenti (stesso tipo di logica).

1.2.1 Sintassi

- **Falso/Assurdo/ \perp (bottom):**
Il simbolo \perp rappresenta la contraddizione, che **non vale mai**. È una proposizione che è sempre falsa.
- **Vero/ \top (top):**
Il simbolo \top rappresenta la verità, che **vale sempre**. È una proposizione che è sempre vera.
- **Predicato Applicato:**
Un predicato applicato, ad esempio “ $2 \leq x$ ”, “ $f(n) = 4$ ”, o “5 è pari”, può **valere o non valere**. Il valore dipende dal predicato e dai suoi argomenti.
- **Congiunzione ($P \wedge Q$):**
La congiunzione $P \wedge Q$ è vera se **entrambi** P e Q sono veri. In altre parole, P e Q **devono essere veri** affinché la formula sia vera.
- **Disgiunzione ($P \vee Q$):**
La disgiunzione $P \vee Q$ è vera se **almeno una** delle formule P o Q è vera. Può essere vera anche se solo una delle proposizioni è vera.
- **Implicazione ($P \Rightarrow Q$):**
L'implicazione $P \Rightarrow Q$ è vera se, **sotto l'ipotesi che P sia vero**, anche Q deve essere vero. Se P è falso, l'implicazione è sempre vera.
- **Negazione ($\neg P$):**
La negazione $\neg P$ è vera se P **implica il falso**. In altre parole, $\neg P$ è vero quando P è falso.
- **Coimplicazione ($P \Leftrightarrow Q$):**
La coimplicazione $P \Leftrightarrow Q$ è vera se P **implica** Q e Q **implica** P ($P \Rightarrow Q \wedge Q \Rightarrow P$). In altre parole, P e Q devono essere veri nelle stesse situazioni.
- **Quantificazione Universale ($\forall x.P$):**
La formula $\forall x.P$ afferma che la proposizione P è vera per **ogni possibile valore di x** . Se P è vera per tutti i valori di x , allora $\forall x.P$ è vero.

- **Quantificazione Esistenziale ($\exists x.P$):**

La formula $\exists x.P$ afferma che **esiste almeno un valore di x** per cui P è vera. Se P è vera per almeno un valore di x , allora $\exists x.P$ è vero.

1.2.2 Connettivi e Quantificatori

- **Connettivi Logici:**

Un connettivo logico costruisce una nuova proposizione a partire da proposizioni già esistenti. La **arietà** di un connettivo è determinata dal numero di proposizioni che prende come input per formare una nuova proposizione.

- **0-ari:**

- * \perp (**Falso**) e \top (**Vero**) sono connettivi **0-ari**, perché non prendono nessuna proposizione come input, ma sono sempre veri o sempre falsi.

- **Unari:**

- * \neg (**Negazione**) è un connettivo **unario**, perché prende una sola proposizione come input per formare la nuova proposizione.

- **Binari:**

- * \wedge (**Congiunzione**), \vee (**Disgiunzione**), \Rightarrow (**Implicazione**) e \Leftrightarrow (**Coimplicazione**) sono connettivi **binari**, perché prendono **due proposizioni** come input per formare una nuova proposizione.

- **Quantificatori:**

I quantificatori \forall e \exists non sono connettivi logici, ma **quantificatori**. Essi agiscono sulla proposizione nel loro ambito e si riferiscono a variabili.

1.2.3 Precedenza e Associatività dei Connettivi

Per ridurre il numero di parentesi nelle formule, si seguono delle regole di **precedenza** e **associatività** per i connettivi logici:

- **Ordine di Precedenza:**

La precedenza dei connettivi logici, che determina l'ordine con cui vengono applicati, è la seguente (dal più alto al più basso):

- \neg (**Negazione**)

- \wedge (**Congiunzione**)

- \vee (**Disgiunzione**)

- \Rightarrow (**Implicazione**) e \Leftrightarrow (**Coimplicazione**)

- **Associatività:** I connettivi \wedge , \vee , \Rightarrow , e \Leftrightarrow sono **associativi a destra**, il che significa che quando ci sono più connettivi dello stesso tipo, vengono raggruppati da destra a sinistra.

Esempio

Consideriamo l'espressione:

$$A \wedge \neg B \vee C \Rightarrow D \Rightarrow B \vee C \vee E$$

Questa formula può essere letta come:

$$((A \wedge (\neg B)) \vee C) \Rightarrow (D \Rightarrow (B \vee (C \vee E)))$$

2 Teoria degli insiemi

La matematica si fonda su entità che possiedono determinate proprietà, le quali sono basate su **assunzioni iniziali**. È fondamentale che queste assunzioni siano **consistenti** tra di loro, ossia che non portino a contraddizioni. Inoltre, è necessario scegliere un **insieme ristretto di entità** che siano in grado di descrivere tutte le altre entità, in modo da costruire una base solida per la teoria matematica.

Alla fine del XIX secolo, la **teoria degli insiemi** emerge come fondamento della matematica. Tuttavia, poco dopo, sorgono problemi, in quanto si scopre che questa teoria presenta delle **inconsistenze**. Per questo motivo, la logica torna a essere una disciplina centrale, proponendo nuovi fondamenti per la matematica.

Nonostante ciò, la teoria degli insiemi resta il fondamento matematico più utilizzato, anche se non è l'unico possibile. Secondo questa teoria, tutto può essere visto come un insieme, e gli insiemi stessi possono contenere altri insiemi, senza che vi sia altro. Questa visione è estremamente efficace nel descrivere entità matematiche come insiemi e consente di trattare concetti complessi, come quelli legati agli infiniti.

Tuttavia, dal punto di vista informatico, la teoria degli insiemi risulta essere poco adatta e inefficiente, presentando diversi svantaggi pratici.

2.1 Teoria degli insiemi naive

La **teoria degli insiemi naive**, formalizzata da Georg Cantor, si basa sull'idea che un insieme sia una **collezione di oggetti** definita in qualunque modo sembri ragionevole farlo. In questa visione, un insieme può contenere qualsiasi tipo di elemento, senza alcuna necessità che questi siano omogenei tra loro. Un insieme viene definito come la **collezione di tutti gli elementi che condividono una stessa proprietà P** , ad esempio $A = \{X \mid P(X)\}$. Gli unici predicati di base utilizzati in questa teoria sono l'**appartenenza** (simbolo \in) e l'**uguaglianza** (simbolo $=$).

Nella teoria degli insiemi naive, un insieme viene visto come una **scatola**: quando “apriamo” una scatola, vediamo gli elementi che contiene, ma non possiamo vedere gli elementi dentro l'insieme di partenza. Inoltre, in questa teoria, l'ordine e le ripetizioni degli elementi non sono rilevanti quando si considera l'uguaglianza tra insiemi.

Tuttavia, la teoria degli insiemi naive si rivela **inconsistente** a causa del famoso **paradosso di Russell**. Questo paradosso si può enunciare come segue:

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

Da questa definizione, si deduce che $X \in X$ se e solo se $X \notin X$, il che porta a una contraddizione logica, perché non può essere vero che X appartenga a se stesso e allo stesso tempo non gli appartenga. Questo paradosso dimostra che la teoria naive degli insiemi porta a **inconsistenza logica**, e perciò si fonda su una base falsa.

Nonostante il paradosso, la teoria degli insiemi rimane estremamente utile in matematica, e quindi i matematici si sono impegnati a trovare un modo per “salvare” la teoria. Una possibile soluzione consiste nel **modificare** l'idea di “insieme” stessa, eliminando l'**assunzione di comprensione**. In particolare, non tutte le collezioni di elementi possono essere considerate insiemi. Alcune collezioni, che non rispettano le regole di questa nuova teoria, vengono chiamate **classi**. Nel caso del paradosso di Russell, X non sarebbe un insieme, ma una classe, che può essere descritta come “la classe degli insiemi che non appartengono a se stessi”. Di conseguenza, la dichiarazione che $X \in X$ è falsa, perché X non è un insieme ma una classe, e quindi non può appartenere a se stessa.

2.2 Teorie assiomatiche degli insiemi

Esistono varie teorie assiomatiche degli insiemi che condividono alcuni aspetti fondamentali:

- ci sono **concetti primitivi** che non vengono definiti (come nella geometria euclidea), perché sarebbe impossibile definire tutto (senza andare all'infinito)
- in particolare i concetti primitivi di partenza sono:
 - **insieme**
 - **appartenenza**
 - **uguaglianza**
- si usano **assiomi**: asseriscono l'esistenza di alcuni insiemi a partire dall'esistenza di altri

Le varie teorie insiemistiche differiscono principalmente riguardo al set di assiomi utilizzati. Una delle più famose/utilizzate è quella di **Zermelo-Fraenkel (ZF)** in quanto è la meno controversa. Bisogna però ricordare che ZF non è mai stata dimostrata **consistente**, ovvero non è dimostrato che non contenga paradossi. Fino ad ora però non ne sono mai stati trovati.

2.3 Teoria di Zermelo-Fraenkel (ZF)

- **Assioma** \rightarrow un'ipotesi che assumiamo \rightarrow siamo interessati solo ai casi in cui vale
- **Definizione** \rightarrow una definizione è un' **abbreviazione** \rightarrow non stiamo ipotizzando nulla di nuovo

2.3.1 Assioma di Estensionalità

L'**Assioma di Estensionalità** afferma che due insiemi sono uguali se e solo se hanno gli stessi elementi:

$$\forall X, \forall Y, (X = Y \iff \forall Z (Z \in X \iff Z \in Y))$$

Due insiemi X e Y sono uguali se, per ogni elemento Z , Z appartiene a X se e solo se appartiene anche a Y . In altre parole, gli insiemi sono uguali se contengono esattamente gli stessi elementi.

2.3.2 Definizione di Sottoinsieme

Un insieme X è un sottoinsieme di un insieme Y se e solo se ogni elemento di X è anche un elemento di Y :

$$X \subseteq Y \stackrel{\text{def}}{=} \forall Z (Z \in X \Rightarrow Z \in Y)$$

L'insieme X è un sottoinsieme di Y se per ogni elemento Z , se Z è un elemento di X , allora Z è anche un elemento di Y . Questo significa che tutti gli elementi di X sono contenuti in Y .

2.3.3 Assioma di separazione

L'**Assioma di separazione** afferma che dato un insieme X , possiamo formare un sottoinsieme Y contenente gli elementi di X che soddisfano una certa proprietà P :

$$\forall X, \exists Y, \forall Z, (Z \in Y \iff Z \in X \wedge P(Z))$$

Possiamo anche scrivere Y come:

$$Y = \{Z \in X \mid P(Z)\}$$

Questo significa che, dato un insieme X e una proprietà P , esiste un insieme Y che contiene esattamente quegli elementi di X che soddisfano P . Possiamo quindi riscrivere l'assioma come:

$$\forall X, \forall Z, (Z \in \{W \in X \mid P(W)\} \iff Z \in X \wedge P(Z))$$

Per utilizzare l'**Assioma di separazione** al fine di riprodurre il **paradosso di Russell**, avremmo bisogno di un insieme U che contenga tutti gli altri insiemi. Ad esempio, consideriamo l'insieme X definito come:

$$X = \{Y \in U \mid Y \notin Y\}$$

Tuttavia, nessun assioma della teoria degli insiemi **ZF** afferma che un tale insieme U esista. La collezione di tutti gli insiemi, infatti, non è un insieme, ma è una **classe propria**. In ZF, tutto ciò che non è un insieme è considerato una **classe**, e le classi proprie non sono trattate come insiemi. Quindi non ricadiamo nel **paradosso di Russel**.

A differenza dell'**assioma di comprensione**, che non partiva da nessun insieme preesistente, l'**assioma di separazione** presuppone già l'esistenza di un insieme da cui creare un sottoinsieme con una proprietà specifica.

2.3.4 Assioma dell'insieme vuoto

L'**Assioma dell'insieme vuoto** afferma che esiste un insieme che non contiene alcun elemento. Formalmente, possiamo scrivere:

$$\exists X, \forall Z, Z \notin X$$

Questo significa che esiste un insieme X tale che nessun elemento Z appartiene a X . Questo insieme viene indicato come l'insieme vuoto, denotato \emptyset . L'assioma può quindi essere riscritto come:

$$\forall Z, Z \notin \emptyset$$

2.3.5 Definizione dell'insieme vuoto

Una volta che abbiamo definito cosa significa un insieme in generale, possiamo usare l'**Assioma di separazione** per definire l'insieme vuoto. In particolare, l'insieme vuoto può essere definito come un sottoinsieme di un insieme arbitrario Y , in cui la proprietà che definisce gli elementi è sempre falsa:

$$\emptyset \stackrel{\text{def}}{=} \{X \in Y \mid \text{false}\}$$

La definizione dell'insieme vuoto è quindi il sottoinsieme di Y che contiene solo gli elementi che soddisfano la condizione falsa. In termini formali:

$$Z \in \{X \in Y \mid \text{false}\} \iff Z \in Y \wedge \text{false} \iff \text{false}$$

Da questa equivalenza, risulta che nessun elemento Z può appartenere all'insieme vuoto, poiché la condizione è sempre falsa. In altre parole, **per ogni Z , Z non appartiene a \emptyset** , il che è esattamente ciò che afferma l'**Assioma dell'insieme vuoto**. (L'assioma sarà ridondante)

2.3.6 Definizione e teorema di intersezione binaria

L'**intersezione** di due insiemi A e B è l'insieme formato da tutti gli elementi che appartengono sia ad A che a B :

$$A \cap B \stackrel{\text{def}}{=} \{X \in A \mid X \in B\}$$

Possiamo inoltre affermare che:

$$X \in A \cap B \iff (X \in A \wedge X \in B)$$

Questo è il **teorema dell'intersezione binaria**, la cui dimostrazione è ovvia per l'**assioma di separazione** e la **definizione di intersezione binaria**.

È possibile estendere il concetto di intersezione binaria a un numero **finito** di insiemi:

$$A_1 \cap A_2 \cap \dots \cap A_n = (\dots ((A_1 \cap A_2) \cap A_3) \dots \cap A_n)$$

Questa è detta **intersezione n-aria** ed è definita **per ricorsione** a partire dall'intersezione binaria. Tuttavia non è possibile, in logica formale, definire, **tramite l'intersezione binaria**, un'intersezione di **infiniti insiemi** con una sola formula, perché una formula logica è sempre **finita**. Scritture informali del tipo:

$$A_1 \cap A_2 \cap A_3 \cap \dots$$

non hanno significato logico rigoroso, poiché i simboli “...” non rappresentano un'entità formale.

2.3.7 Definizione di intersezione

Quando si vuole intersecare un **numero arbitrario di insiemi**, (anche potenzialmente infinito), si procede raccogliendo tutti gli insiemi da intersecare in un **insieme di insiemi**, che indichiamo con F .

In questo modo, possiamo definire l'intersezione di tutti gli insiemi appartenenti a F .

Non esiste però un **elemento neutro** per l'operazione di intersezione.

Infatti, l'insieme che dovrebbe fungere da elemento neutro sarebbe l'insieme “universale” contenente tutti gli insiemi — ma un tale oggetto non esiste come insieme, bensì come **classe propria** (troppo grande per essere un insieme). Per questo motivo, si adotta la seguente convenzione:

$$\bigcap F \stackrel{\text{def}}{=} \emptyset \quad \text{se } F = \emptyset$$

ovvero, l'intersezione dell'insieme vuoto di insiemi è definita come l'insieme vuoto.

In generale, dato un insieme F i cui elementi sono insiemi, si definisce la **loro intersezione** come:

$$\bigcap F \stackrel{\text{def}}{=} \begin{cases} \emptyset & \text{se } F = \emptyset \\ \{ X \in A \mid \forall Y, (Y \in F \Rightarrow X \in Y) \} & \text{se } F \neq \emptyset \end{cases}$$

dove $A \in F$ è un qualunque elemento di F .

La definizione **non dipende dalla scelta di A** .

Detto F l'insieme (eventualmente infinito) di tutti gli insiemi da intersecare, l'**insieme intersezione** Y di tutti quelli si può anche scrivere:

$$\bigcap F = \bigcap_{Y \in F} Y$$

che si legge “**intersezione di tutti gli insiemi Y appartenenti a F** ”.

Esempio informale

Se $F = \{ A, B, C \}$, allora: $\bigcap_{Y \in \{A, B, C\}} Y = A \cap B \cap C$

2.3.8 Assioma dell'unione

L'**unione** è un'operazione potenzialmente “pericolosa” dal punto di vista assiomatico, perché permette di “raccolgere” in un unico insieme tutti gli elementi appartenenti ad altri insiemi.

Per garantire che tale operazione produca effettivamente un **insieme ben definito**, si introduce l'**assioma dell'unione**.

Dato un insieme F i cui elementi sono a loro volta insiemi, esiste un insieme che contiene **tutti e soli** gli elementi appartenenti ad almeno uno degli insiemi di F . In altre parole, possiamo definire:

$$\forall F, \exists X, \forall Z, (Z \in X \iff \exists Y, (Y \in F \wedge Z \in Y))$$

L'insieme X garantito da questo assioma si chiama **unione di F** e si indica con:

$$\bigcup F \quad \text{o anche} \quad \bigcup_{Y \in F} Y$$

2.3.9 Unione binaria (teorema)

Il teorema dell'**unione binaria** afferma che, dati due insiemi A e B , **esiste sempre** un insieme X i cui elementi sono **tutti e soli** gli elementi appartenenti ad almeno uno dei due insiemi:

$$\forall A, \forall B, \exists X, \forall Z, (Z \in X \iff Z \in A \vee Z \in B)$$

Questo insieme X si chiama **unione binaria** di A e B .

Per il momento non è dimostrabile, infatti non è ancora possibile dimostrare per ogni A e B l'esistenza di un insieme che contiene **solo** A e B .

Scrivendo X come $A \cup B$, possiamo esprimere il teorema nella forma più usuale:

$$\forall A, \forall B, \forall Z, (Z \in A \cup B \iff Z \in A \vee Z \in B)$$

2.3.10 Assioma del singoletto

L'**Assioma del Singoletto** garantisce l'esistenza di un insieme di un solo elemento, in particolare dice che:

$$\forall X, \exists Y, \forall Z, (Z \in Y \iff Z = X)$$

Indicando l'insieme Y come $\{X\}$ (ovvero quell'insieme che contiene solo X), è possibile riscrivere l'assioma come:

$$\forall X, \forall Z, (Z \in \{X\} \iff Z = X)$$

Anche questo assioma è **in realtà ridondante**, in quanto si potrà dimostrare in seguito a partire dall'assioma di rimpiazzamento.

Aggiungendo questo assioma possiamo introdurre anche un "abuso" di notazione, ovvero possiamo indicare con $\{A_1, \dots, A_n\}$ l'insieme $\{A_1\} \cup \dots \cup \{A_n\}$, che esiste in virtù degli assiomi del singoletto e dell'unione:

$$X \in \{A_1, \dots, A_n\} \iff X = A_1 \vee \dots \vee X = A_n$$

A partire da questo assioma è possibile ora formare nuovi tipi di insiemi (in quanto fin ora l'unico insieme di cui era garantita l'esistenza era l'insieme vuoto):

- Un insieme che contiene solo l'insieme vuoto (singoletto del vuoto): $\{\emptyset\}$
- Un insieme che contiene solo il singoletto del vuoto: $\{\{\emptyset\}\}$
- E così via...

Usando poi l'unione posso creare altri insiemi ancora:

- Un insieme ottenuto dall'unione dell'insieme vuoto e del singoletto del singoletto del vuoto: $\{\emptyset, \{\emptyset\}\}$
- Un insieme ottenuto dall'unione del singoletto del vuoto e del singoletto del singoletto del vuoto: $\{\{\emptyset\}, \{\{\emptyset\}\}\}$
- E così via

Notiamo quindi come è possibile formare **un'infinità** di insiemi, per ora però **tutti finiti**.

2.3.11 Costruzione dei numeri naturali

Per “costruire” i numeri naturali, dobbiamo prima introdurre il concetto di numero. Tuttavia, non possiamo farlo direttamente con questa logica, quindi dobbiamo ricorrere alla **meta-logica**, utilizzando il concetto di **meta-numero naturale**.

Indicheremo l'utilizzo della meta-logica con le parentesi quadre doppie $[[\cdot]]$, per differenziarlo dalla logica formale.

- Lo zero è rappresentato dall'insieme vuoto:

$$[[0]] \stackrel{\text{def}}{=} \emptyset$$

- Il successore di un numero naturale n (cioè $n+1$) è rappresentato dall'insieme che contiene tutti gli elementi di n e in più il singoletto di n :

$$[[n+1]] \stackrel{\text{def}}{=} [[n]] \cup \{[[n]]\}$$

- Denoteremo questi insiemi con la notazione di numeri naturali:

$$[[0]] = \emptyset = 0$$

$$[[1]] = \{\emptyset\} = 1$$

$$[[2]] = \{\emptyset, \{\emptyset\}\} = 2$$

$$[[3]] = \{\emptyset, \{\emptyset\}, \{\emptyset, \{\emptyset\}\}\} = 3$$

Questa costruzione ci permette di rappresentare i numeri naturali come insiemi, dove ogni insieme contiene tutti gli insiemi precedenti. In particolare è una rappresentazione “comoda”, in quanto possiamo definire facilmente alcune operazioni sui numeri naturali.

Esempi di definizioni

- La relazione di ordine \leq tra numeri naturali è definita come:

$$n \leq m := n \in m \text{ (o anche } n \subseteq m \text{)}$$

- Il massimo tra due numeri naturali n e m è definito come:

$$\max\{n, m\} := n \cup m$$

- Il minimo tra due numeri naturali n e m è definito come:

$$\min\{n, m\} := n \cap m$$

Sarà poi necessario dimostrare che questi numeri naturali, così definiti, siano effettivamente tutti diversi. Per il momento ci limitiamo a dimostrare che presi due numeri naturali distinti essi sono diversi.

Dimostrazione che due numeri sono diversi

Se vogliamo per esempio dimostrare che $1 \neq 2$ procederemo così:

Supponiamo che $1 = 2$.

Quindi per l'assioma di estensionalità dimostriamo che $\forall Z, Z \in 1 \iff Z \in 2$. (H_1).

Per l'assioma dell'unione, l'assioma del singoletto e la costruzione dei numeri naturali dimostriamo che $1 \in 2$.

Quindi per H_1 dimostriamo che $1 \in 1$.

Quindi per la costruzione dei numeri naturali e per l'assioma del singoletto dimostriamo che $\{\emptyset\} = 1$ e $1 = \emptyset$.

Quindi per l'assioma di estensionalità e per la transitività dell'uguaglianza dimostriamo che $\forall Z, Z \in \{\emptyset\} \iff Z \in \emptyset$.

Quindi, poiché $\emptyset \in \{\emptyset\}$ per l'assioma del singoletto, dimostriamo che $\emptyset \in \emptyset$.

Quindi abbiamo un assurdo per l'assioma del vuoto.

Quindi $1 \neq 2$.

Qed.

2.3.12 Assioma dell'infinito

L'assioma dell'infinito afferma che esiste un insieme che contiene almeno tutti (gli encoding de) i numeri naturali:

$$\exists Y, (\emptyset \in Y \wedge \forall N, (N \in Y \Rightarrow N \cup \{N\} \in Y))$$

Indicheremo temporaneamente con \mathcal{N} tale insieme.

Più avanti, quando capiremo cos'è l'infinito, sarà possibile dimostrare che questo è l'unico insieme infinito visto fin ora, e che è proprio grazie a questo insieme/assioma che esistono gli insiemi infiniti (costruiti a partire da questo).

2.3.13 Teorema dell'esistenza di \mathbb{N}

Combinando l'assioma dell'infinito e altri non ancora visti si arriva a dimostrare l'esistenza di \mathbb{N} , che contiene tutti e solo gli encoding dei numeri naturali.

2.3.14 Assioma dell'insieme potenza

L'assioma dell'insieme potenza afferma che esiste l'insieme dei sottoinsiemi di un insieme dato:

$$\forall X, \exists Y, \forall Z, (Z \in Y \iff Z \subseteq X)$$

Indicheremo Y come 2^X (insieme potenza) oppure come $\mathcal{P}(X)$ (insieme delle parti).

Esempio

$$2^{\{1,2\}} = \{\emptyset, \{1\}, \{2\}, \{1, 2\}\}$$

È importante notare come, su insiemi finiti, sarebbe comunque possibile costruire il relativo insieme delle parti; il problema si presenterebbe invece per gli insiemi infiniti (come per \mathbb{N}). Questo assioma invece garantisce l'esistenza dell'insieme delle parti anche per gli insiemi infiniti.

Questo sarà ad esempio fondamentale nella costruzione dei numeri reali (a partire dall'insieme delle parti di \mathbb{N}).

2.3.15 Assioma di regolarità (o di fondazione)

Se un insieme è non vuoto ha almeno un elemento con il quale è **disgiunto** (non ha un'intersezione).

In pratica assicura che gli insiemi possano contenere solo cose "più piccole" e non possono quindi contenere sé stessi (ricorsivamente).

Come conseguenza di questo ha quindi senso cercare di misurare e definire la cardinalità (taglia) di un insieme.

2.3.16 Assioma di rimpiazzamento

Intuitivamente l'assioma di rimpiazzamento afferma che l'immagine di un insieme rispetto a una formula che descrive una funzione è ancora un insieme.

Ovvero, sempre intuitivamente: se A è un insieme, quindi è abbastanza piccolo, e a ogni elemento ne associo un altro, in una relazione molti-a-uno, quello che ottengo come immagine è ancora piccolo (un insieme).

3 Regole di dimostrazione

- **Enunciato:** ciò che vogliamo dimostrare, costituito da un insieme di ipotesi e da una conclusione
- Tutti gli assiomi possono essere sempre usati come ipotesi (sono ipotesi universali)
- Procediamo nella dimostrazione in due modi, seguendo le regole di introduzione ed eliminazione:
 - **Introduzione:** il connettivo non è presente nelle premesse, ma appare nella conclusione, procediamo quindi manipolando la conclusione
 - **Eliminazione:** il connettivo è presente in una delle premesse (ipotesi o risultato intermedio), ma scompare nella conclusione; procediamo quindi manipolando le premesse/ipotesi

3.1 \forall -Introduzione

La regola di \forall -**Introduzione** afferma che per dimostrare una proposizione che coinvolge un quantificatore universale $\forall x.P(x)$ (ovvero, “per ogni x , vale $P(x)$ ”), bisogna assumere un x arbitrario (**insieme fissato**) e dimostrare che $P(x)$ è vero per tale x . Per insieme fissato si intende un insieme generico, di cui non si cambierà nulla e non si assumerà nulla: se si dimostra qualcosa per un x fissato, vuol dire che si può ripetere la dimostrazione per qualunque altro x' .

Esempio

“Sia X (un insieme) fissato; ...”
(i “...” rappresentano la prova di $P(X)$.)

Esempio in Lean

```
1 assume X: set
2 -- prova che P(x) è vero per questo x
```

3.2 \forall -Eliminazione

La regola di \forall -**Eliminazione** permette di concludere che $P(x)$ è vero per un valore qualunque di x , poiché sappiamo che $\forall x.P(x)$ è vero per ogni x . Da un'ipotesi che afferma $\forall x.P(x)$, possiamo concludere che $P(a)$ è vero per un valore specifico a .

Esempio in Lean

```
1 by NOME_IPOTESI we proved CONCLUSIONE as NOME_NUOVO_RISULTATO
```

3.3 \Rightarrow -Introduzione

La regola di \Rightarrow -**Introduzione** afferma che per dimostrare una proposizione implicativa $P \Rightarrow Q$ (ovvero, “se P allora Q ”), bisogna assumere P come ipotesi e, dopo aver dimostrato che Q segue da P , si conclude che $P \Rightarrow Q$ è vero.

Esempio

“Suppongo P ($NOME_IPOTESI$); ...”
(i “...” rappresentano la prova di Q partendo dall'ipotesi P).

Esempio in Lean

```
1 suppose P as NOME_IPOTESI,
2 -- prova che Q segue da P
```

3.4 \Rightarrow -Eliminazione

La regola di \Rightarrow -Eliminazione afferma che se sappiamo che $P \Rightarrow Q$ (da un'ipotesi o da un risultato intermedio) e sappiamo che P è vero (da un'altra ipotesi o risultato intermedio), possiamo concludere che Q è vero.

Esempio in Lean

```
1 "by NOME_IPO_PQ, NOME_IPO_P we proved Q as NOME_RISULTATO_INTERMEDIO"
```

3.5 \Rightarrow -Eliminazione (variante)

La regola di \Rightarrow -Eliminazione (variante) afferma che, dato un risultato intermedio o un'ipotesi del tipo $P \Rightarrow Q$ (denominata H), se si vuole concludere Q , si può procedere dicendo: “per H , per dimostrare Q mi posso ridurre a dimostrare P ”. In altre parole, si sfrutta l'ipotesi H per ridurre la dimostrazione di Q alla dimostrazione di P .

Esempio

“Per H , per dimostrare Q , mi posso ridurre a dimostrare P ; ...”
(i “...” rappresentano la prova di P che, una volta dimostrata, porterà alla conclusione di Q).

Esempio in Lean

```
1 by H it suffices to prove P
```

3.6 \Leftrightarrow -Introduzione

La regola di \Leftrightarrow -Introduzione afferma che per dimostrare una bicondizionale $P \Leftrightarrow Q$ (ovvero, “ P se e solo se Q ”), bisogna dimostrare entrambe le implicazioni: $P \Rightarrow Q$ e $Q \Rightarrow P$.

Bisognerà quindi:

1. Dimostrare $P \Rightarrow Q$.
2. Dimostrare $Q \Rightarrow P$.

Esempio

“Per dimostrare $P \Leftrightarrow Q$, dimostriamo che $P \Rightarrow Q$ e poi che $Q \Rightarrow P$.”

Esempio in Lean

```
1 we split the proof
2 . we need to prove P → Q
3 -- Dimostrazione
4 . we need to prove Q → P
5 -- Dimostrazione
```

3.7 \Leftrightarrow -Eliminazione

La regola di \Leftrightarrow -Eliminazione afferma che un'ipotesi $P \Leftrightarrow Q$ può essere utilizzata sia come un'ipotesi $P \Rightarrow Q$, che come un'ipotesi $Q \Rightarrow P$, a seconda della necessità nella dimostrazione.

Esempio

Ipotesi $H: P \iff Q$
“Per H dimostriamo $P \Rightarrow Q$ ”
Oppure
“Per H dimostriamo $Q \Rightarrow P$ ”

3.8 Abbreviazioni

- **Per ogni tale che:**
“Sia x (un insieme) fissato; suppongo $P(x)$; ...” si può abbreviare con: “**sia x tale che $P(x)$** ”. Questa formula si usa per dimostrare $\forall x. (P(x) \Rightarrow Q(x))$.
- **Da H_1, \dots, H_n :**
Se si ha un insieme di ipotesi H_1, \dots, H_n , con ognuna della forma $\forall \vec{x}. (Q_{i1}(\vec{x}) \Rightarrow \dots \Rightarrow Q_{in}(\vec{x}))$, l’abbreviazione usata è: “**Da H_1, \dots, H_n ho $P(H)$** ”.
Questa espressione indica l’applicazione di **un numero arbitrario di regole di eliminazione**, partendo dalle ipotesi H_1, \dots, H_n , fino a giungere alla conclusione P .

In Lean diventa:

```
1 by H1, ..., Hn we proved P as H
```

- **Uso di “quindi” e sinonimi:**
“**Quindi**” e termini simili vengono usati come riferimenti all’ultima ipotesi o risultato intermedio, anche senza citare esplicitamente il nome.

In Lean:

```
1 thus ...
```

3.9 Espansione di definizioni

Quando si espande una definizione, si usa l’espressione “ P , ovvero Q ” per indicare che la proposizione P è equivalente a Q .

Esempio

Se abbiamo una definizione come $A \subseteq B$, possiamo espanderla come $\forall X (X \in A \Rightarrow X \in B)$
Diremo quindi:
“ $A \subseteq B$ ovvero $\forall X. (X \in A \Rightarrow X \in B)$.”

Esempio in Lean

```
1 A ⊆ B that is equivalent to ∀X, X ∈ A → X ∈ B
```

3.10 Esplicitazione della conclusione

A volte si esplicita ciò che dobbiamo dimostrare \rightarrow perché potrebbe cambiare nel corso della dimostrazione.

Esempio

Dobbiamo dimostrare P .

Esempio in Lean

```
1 we need to prove P
```

3.11 Ovvio

“Ovvio” si usa quando il lettore può ricostruire la prova per conto suo (ad esempio combinando le ipotesi elencate).

Esempio in Lean

```
1 done
```

3.12 Assurdo-Eliminazione

Dalle regole di eliminazione delle ipotesi possiamo ottenere il **bottom** (\perp) (altrimenti non sarebbe possibile dimostrarlo).

Se siamo in un caso che non si verifica mai, allora possiamo concludere qualunque cosa: se abbiamo dimostrato l'**assurdo** possiamo dimostrare **qualsiasi proposizione**.

Esempio

Per ... dimostro l'assurdo. Quindi [quello che voglio].

Esempio in Lean

```
1 by ... we proved False
2 thus ...
```

3.13 \neg -Introduzione

$\neg P$ è un'abbreviazione di $P \Rightarrow \perp$, quindi per dimostrare $\neg P$ si assume P e si dimostra l'assurdo.

Importante: Questa non è una dimostrazione per assurdo!

Esempio

Suppongo P (NOME_IPOTESI); ...
(i “...” rappresentano la prova dell'assurdo.)

3.14 \neg -Eliminazione

Se ho un'ipotesi $\neg P$ la posso combinare con un'altra ipotesi che afferma P : si conclude così l'**assurdo**.

Esempio

- $H_1: P$
- $H_2: \neg P$
“Da H_1 e H_2 dimostro l'assurdo. Quindi, ovvio.”

3.15 \wedge -Introduzione

La regola di \wedge -Introduzione ci permette di procedere in due modi per dimostrare $P \wedge Q$:

1. **Spezzando la prova:** si dimostra prima P poi Q .

Esempio

“Per dimostrare $P \wedge Q$, dimostriamo P e poi Q .”

Esempio in Lean

```
1 we split the proof
2 . we need to prove P
3 -- Dimostrazione di P
4 . we need to prove Q
5 -- Dimostrazione di Q
```

2. **Dalle prove già ottenute di P e Q :** si conclude direttamente $P \wedge Q$.

Esempio

$H_1: P$
 $H_2: Q$
“Da H_1, H_2 dimostriamo $P \wedge Q$.”

3.16 \wedge -Eliminazione

La regola di \wedge -Eliminazione ci permette, quando abbiamo un risultato intermedio in cui vale $P \wedge Q$, di spezzare tale ipotesi in due ipotesi diverse:

- una per cui vale P
- una per cui vale Q

In alternativa posso anche usare la stessa ipotesi $P \wedge Q$ sia per concludere P , che per concludere Q .

Esempio in Lean

```
1 by NOME_p_and_q we proved P as H1 and Q as H2
```

3.17 \vee -Introduzione

La regola di \vee -Introduzione ci permette, quando dobbiamo dimostrare $P \vee Q$, di ridurci a dimostrare o P o Q , dichiarandolo.

Esempio

“Dimostro P ” oppure “Dimostro Q ”

Se ho già una dimostrazione di P o una di Q , ho già dimostrato $P \vee Q$

Esempio in Lean

```
1 by NOME_p we proved P ∨ Q
```

Può però capitare di trovarsi in un punto della dimostrazione in cui non si sa ancora se scegliere P o Q : in tali casi bisogna aspettare.

3.18 \vee -Eliminazione

Data un'ipotesi $P \vee Q$, la regola di \vee -Eliminazione ci permette di spezzare la dimostrazione in **due casi**:

- i casi in cui vale P .
- i casi in cui vale Q .

Bisogna quindi dimostrare per entrambi i casi. Ogni sotto dimostrazione avrà a disposizione un'ipotesi in più (ad esempio nel caso in cui valga P , avrò l'ipotesi P).

Le ipotesi di un caso non influenzano quelle dell'altro.

Esempio

*“Procedo per casi:
- Caso in cui valga P .
Suppongo $P(H)$
- Caso in cui valga Q .
Suppongo $Q(H)$”*

Esempio in Lean

```
1 we proceed by cases on NOME_P_or_Q to prove CONCLUSIONE
2 . case or_introl
3   suppose P as H
4   ...
5 . case or_intror
6   suppose Q as H
7   ...
```

3.19 Risultati intermedi

Possiamo usare risultati intermedi per concludere **nuovi risultati intermedi**, diversi dalla conclusione corrente. Combiniamo quindi varie ipotesi per ottenere una nuova ipotesi che potremo utilizzare.

Esempio

$H_1: A$
 $H_2: B$
“Per H_1 e H_2 si ha $A \wedge B$ (H_3)”

3.20 \exists -Introduzione

$\exists x. P(x)$ significa che P vale se esiste almeno un x per cui vale, quindi dimostro che vale per uno **specifico elemento** che scelgo. Il valore di x da scegliere non andrà preso a caso, ma dopo aver capito/visto quale sia il valore migliore per la dimostrazione. All'inizio potrà non essere chiaro quale, o proprio non esserci un valore concreto da poter scegliere.

Esempio

*“Scelgo E e dimostro $P(E)$”**

Esempio in Lean

```
1 we choose E ...
```

3.21 \exists -Eliminazione

Applichiamo la regola di \exists -**Eliminazione** in quelle situazioni in cui $\exists x. P(x)$: ovvero quando esiste un valore di x per il quale vale P , ma non possiamo scegliere una x a caso. E non possiamo neanche dimostrare per tutti i casi possibili (che potrebbero essere infiniti). Si procede quindi scegliendo un x generico di cui non si sa niente, se non che $P(x)$. Se la dimostrazione funziona vuol dire che è abbastanza generica da funzionare per qualunque altro x' per cui valga $P(x')$.

Importante: x deve essere una variabile **NON** già in uso.

Esempio

“Sia X tale che $P(x)$ (H)”

Esempio in Lean

```
1 by PROOF_EXISTS_x_Px let x: set such that P(x) as H
```

4 Teoremi e Dimostrazioni

4.1 Teorema di Riflessività del \subseteq

Il Teorema di Riflessività del \subseteq afferma che ogni insieme è sempre sottoinsieme di sé stesso:

$$X \subseteq X$$

Interpretazione: Ogni insieme è sottoinsieme di sé stesso. Questo significa che ogni elemento di X è automaticamente anche un elemento di X , quindi X è sempre sottoinsieme di X .

Dimostrazione

Sia X un insieme fissato.
Dobbiamo dimostrare che $X \subseteq X$, ovvero: $\forall Y. Y \in X \Rightarrow Y \in X$.
Sia Y un insieme fissato.
Supponiamo $Y \in X$ (ipotesi H).
Dobbiamo dimostrare $Y \in X$.
Ovvio per l'ipotesi H .
Qed.

Dimostrazione in Lean

```
1 theorem reflexivity_inclusion:  $\forall A, A \subseteq A$  := by
2   assume A: set
3   we need to prove  $A \subseteq A$  that is equivalent to  $\forall Z, Z \in A \rightarrow Z \in A$ 
4   assume Z: set
5   suppose  $Z \in A$  as H
6   we need to prove  $Z \in A$ 
7   by H done
```

4.2 Teorema di Anti-simmetria del \subseteq

Il Teorema dell'Anti-simmetria del \subseteq afferma che:

$$X \subseteq Y \text{ e } Y \subseteq X \Rightarrow X = Y$$

Interpretazione: Se due insiemi sono reciprocamente sottoinsiemi l'uno dell'altro, significa che contengono esattamente gli stessi elementi, quindi sono uguali. In altre parole, X è uguale a Y se ogni elemento di X è in Y e ogni elemento di Y è in X .

Dimostrazione

Siano X e Y insiemi fissati.
Supponiamo $X \subseteq Y$, ovvero $\forall Z. Z \in X \Rightarrow Z \in Y$ (ipotesi H_1).
Supponiamo $Y \subseteq X$, ovvero $\forall Z. Z \in Y \Rightarrow Z \in X$ (ipotesi H_2).
Dobbiamo dimostrare $X = Y$.
Per l'**Assioma di Estensionalità**, è sufficiente dimostrare $\forall Z. Z \in X \Leftrightarrow Z \in Y$.
Sia Z un insieme fissato.
Dividiamo la dimostrazione:
- Dobbiamo dimostrare $Z \in X \Rightarrow Z \in Y$ Supponiamo $Z \in X$ (ipotesi H_3).
Dobbiamo dimostrare $Z \in Y$.
Ovvio per H_1, H_3 .
- Dobbiamo dimostrare $Z \in Y \Rightarrow Z \in X$.
Supponiamo $Z \in Y$ (ipotesi H_4).
Dobbiamo dimostrare $Z \in X$.
Ovvio per H_2, H_4 .
Qed.

Dimostrazione in Lean

```

1 theorem subset_to_eq:  $\forall A B, A \subseteq B \rightarrow B \subseteq A \rightarrow A = B$  := by
2   assume A: set
3   assume B: set
4   suppose  $A \subseteq B$  that is equivalent to  $\forall Z, Z \in A \rightarrow Z \in B$  as H1
5   suppose  $B \subseteq A$  that is equivalent to  $\forall Z, Z \in B \rightarrow Z \in A$  as H2
6   we need to prove  $A = B$ 
7   by ax_extensionality1 it suffices to prove  $\forall Z, Z \in A \leftrightarrow Z \in B$ 
8   assume Z: set
9   we split the proof
10  . we need to prove  $Z \in A \rightarrow Z \in B$ 
11    suppose  $Z \in A$  as H3
12    we need to prove  $Z \in B$ 
13    by H1, H3 done
14  . we need to prove  $Z \in B \rightarrow Z \in A$ 
15    suppose  $Z \in B$  as H3
16    we need to prove  $Z \in A$ 
17    by H2, H3 done

```

4.3 Teorema di Transitività del \subseteq

Il Teorema di Transitività del \subseteq afferma che:

$$X \subseteq Y \quad \text{e} \quad Y \subseteq Z \quad \Rightarrow \quad X \subseteq Z$$

Interpretazione: Se un insieme X è sottoinsieme di un insieme Y , e Y è a sua volta sottoinsieme di un insieme Z , allora X è anche sottoinsieme di Z . Questo teorema esprime la proprietà di transizione tra inclusioni di insiemi.

Dimostrazione

Siano X, Y e Z insiemi fissati.
 Supponiamo $X \subseteq Y$, ovvero $\forall A. A \in X \Rightarrow A \in Y$ (ipotesi H_1).
 Supponiamo $Y \subseteq Z$, ovvero $\forall A. A \in Y \Rightarrow A \in Z$ (ipotesi H_2).
 Dobbiamo dimostrare $X \subseteq Z$, ovvero $\forall A. A \in X \Rightarrow A \in Z$.
 Sia A un insieme fissato.
 Supponiamo $A \in X$ (ipotesi H_3).
 Dobbiamo dimostrare $A \in Z$.
 Per H_1, H_3 mostriamo che $A \in Y$ (ipotesi H_4).
 Per H_2, H_4 mostriamo che $A \in Z$.
 Quindi ovvio.
Qed.

Dimostrazione in Lean

```

1 theorem transitivity_inclusion:  $\forall A B C, A \subseteq B \rightarrow B \subseteq C \rightarrow A \subseteq C$  := by
2   assume A : set
3   assume B : set
4   assume C : set
5   suppose  $A \subseteq B$  that is equivalent to  $\forall Z, Z \in A \rightarrow Z \in B$  as H1
6   suppose  $B \subseteq C$  that is equivalent to  $\forall Z, Z \in B \rightarrow Z \in C$  as H2
7   we need to prove  $A \subseteq C$  that is equivalent to  $\forall Z, Z \in A \rightarrow Z \in C$ 
8   assume Z : set
9   suppose  $Z \in A$  as H3
10  by H1, H3 we proved  $Z \in B$  as H4
11  by H2, H4 we proved  $Z \in C$ 
12  thus done

```

4.4 Teorema del Vuoto come sottoinsieme di ogni cosa

Il teorema afferma che l'insieme vuoto \emptyset è sottoinsieme di ogni insieme:

$$\emptyset \subseteq X$$

Interpretazione: L'insieme vuoto è considerato un **sottoinsieme di ogni insieme**, poiché non contiene elementi. La condizione per essere sottoinsieme è che ogni elemento dell'insieme vuoto sia anche un elemento di X , ma non essendoci elementi nell'insieme vuoto, questa condizione è automaticamente soddisfatta per qualsiasi insieme X .

Dimostrazione

Sia X un insieme fissato.

Dobbiamo dimostrare $\emptyset \subseteq X$ ovvero $\forall Z. Z \in \emptyset \Rightarrow Z \in X$.

Sia Z un insieme fissato.

Supponiamo $Z \in \emptyset$ (ipotesi H_1).

Dobbiamo dimostrare $Z \in X$.

Per l'**Assioma dell'Insieme Vuoto** sappiamo che $\neg(Z \in \emptyset)$ (ipotesi H_2).

Per H_1, H_2 abbiamo un assurdo.

Quindi dimostriamo che $Z \in X$.

Quindi ovvio.

Qed.

Dimostrazione in Lean

```
1 theorem emptyset_is_subset:  $\forall A, \emptyset \subseteq A$  := by
2   assume A : set
3   we need to prove  $\emptyset \subseteq A$  that is equivalent to  $\forall Z, Z \in \emptyset \rightarrow Z \in A$ 
4   assume Z : set
5   suppose  $Z \in \emptyset$  as H1
6   we need to prove  $Z \in A$ 
7   by ax_empty we proved  $\neg(Z \in \emptyset)$  as H2
8   by H1, H2 we proved False
9   thus we proved  $Z \in A$ 
10  thus done
```

4.5 Teorema del vuoto come unico sottoinsieme del vuoto

Il teorema afferma che l'unico sottoinsieme dell'insieme vuoto è l'insieme vuoto stesso:

$$X \subseteq \emptyset \Rightarrow X = \emptyset$$

Interpretazione: L'insieme vuoto **non contiene alcun elemento**, quindi non esiste alcun insieme non vuoto che possa essere suo sottoinsieme.

L'unico insieme che soddisfa la condizione "ogni elemento di X appartiene anche a \emptyset " è proprio $X = \emptyset$. In altre parole, il vuoto è sottoinsieme solo di sé stesso.

Dimostrazione

Sia X un insieme tale che $X \subseteq \emptyset$, ovvero $\forall Z, (Z \in X \Rightarrow Z \in \emptyset)$ (ipotesi H).

Dobbiamo dimostrare $X = \emptyset$.

Per l'**assioma di estensionalità** possiamo ridurci a dimostrare $\forall Z, (Z \in X \Leftrightarrow Z \in \emptyset)$.

Sia Z un insieme.

Dividiamo la dimostrazione:

- Dobbiamo dimostrare $Z \in X \Rightarrow Z \in \emptyset$.

Per H ovvio.

- Dobbiamo dimostrare $Z \in \emptyset \Rightarrow Z \in X$.

Per il **teorema del vuoto come sottoinsieme di ogni cosa** dimostriamo che $\forall Z, (Z \in \emptyset \Rightarrow Z \in X)$.

Quindi ovvio.

Qed.

Dimostrazione alternativa

Sia X un insieme tale che $X \subseteq \emptyset$ (ipotesi H).

Dobbiamo dimostrare $X = \emptyset$.

Per il **teorema del vuoto come sottoinsieme di ogni cosa** dimostriamo che $\emptyset \subseteq X$.

Quindi, per H e il **teorema dell'anti-simmetria del \subseteq** , ovvio.

Qed.

Dimostrazione in Lean

```
1 theorem subseq_emptyset:  $\forall X, X \subseteq \emptyset \rightarrow X = \emptyset$  := by
2   assume X : set
3   suppose X  $\subseteq$   $\emptyset$  as H
4   we need to prove X =  $\emptyset$ 
5   by emptyset_is_subset we proved  $\emptyset \subseteq X$ 
6   thus by H, subset_to_eq done
```

4.6 Teorema dell'intersezione con il vuoto

Il teorema afferma che l'intersezione di un qualunque insieme X con l'insieme vuoto è l'insieme vuoto stesso:

$$X \cap \emptyset = \emptyset$$

Interpretazione: Poiché l'insieme vuoto non contiene elementi, non esistono elementi comuni tra X e \emptyset . L'intersezione risulta quindi vuota.

Dimostrazione

Sia X un insieme.

Per l'**assioma di estensionalità** possiamo ridurci a dimostrare $\forall Z, (Z \in X \cap \emptyset \Leftrightarrow Z \in \emptyset)$.

Sia Z un insieme. Dividiamo la dimostrazione:

- Dobbiamo dimostrare $Z \in X \cap \emptyset \Rightarrow Z \in \emptyset$.

Supponiamo $Z \in X \cap \emptyset$.

Quindi, per il **teorema dell'intersezione binaria**, $Z \in X$ (ipotesi H_1) e $Z \in \emptyset$ (ipotesi H_2).

Per H_2 ovvio.

- Dobbiamo dimostrare $Z \in \emptyset \Rightarrow Z \in X \cap \emptyset$.

Supponiamo $Z \in \emptyset$.

Per il **teorema del vuoto come sottoinsieme di ogni cosa**, ovvio.

Qed.

Dimostrazione in Lean

```
1 theorem intersect_empty:  $\forall A, A \cap \emptyset = \emptyset$  := by
2   assume A : set
3   by ax_extensivity1 it suffices to prove  $\forall Z, Z \in A \cap \emptyset \Leftrightarrow Z \in \emptyset$ 
4   assume Z : set
5   we split the proof
6   . we need to prove  $Z \in A \cap \emptyset \rightarrow Z \in \emptyset$ 
7     suppose Z  $\in$   $A \cap \emptyset$ 
8     thus by ax_intersect1 we proved Z  $\in$  A as H1 and Z  $\in$   $\emptyset$  as H2
9     by H2 done
10  . we need to prove  $Z \in \emptyset \rightarrow Z \in A \cap \emptyset$ 
11    suppose Z  $\in$   $\emptyset$ 
12    thus by emptyset_is_subset done
```

4.7 Teorema della Monotonia dell'intersezione

Per ogni insieme X, X', Y vale che:

$$X \subseteq X' \Rightarrow X \cap Y \subseteq X' \cap Y$$

Interpretazione: Se X è contenuto in X' , allora anche la parte di X che si trova in Y sarà contenuta nella parte di X' che si trova in Y . In altre parole, l'intersezione è **monotona** rispetto all'inclusione: aumentare uno degli insiemi non può ridurre l'intersezione.

Dimostrazione

Siano X, X' e Y insiemi tali che $X \subseteq X'$, ovvero $\forall Z, (Z \in X \Rightarrow Z \in X')$ (ipotesi H_1).
Dobbiamo dimostrare $X \cap Y \subseteq X' \cap Y$, ovvero $\forall Z, (Z \in X \cap Y \Rightarrow Z \in X' \cap Y)$.
Sia Z un insieme tale che $Z \in X \cap Y$.
Quindi, per il **teorema dell'intersezione binaria**, $Z \in X$ (ipotesi H_2) e $Z \in Y$ (ipotesi H_3).
Dobbiamo dimostrare $Z \in X' \cap Y$.
Per il **teorema dell'intersezione binaria** possiamo ridurci a dimostrare $Z \in X' \wedge Z \in Y$.
Per H_1 e H_2 dimostriamo $Z \in X'$.
Quindi, per H_3 , ovvio.
Qed.

Dimostrazione in Lean

```
1 theorem intersect_monotone:  $\forall A B A', \subseteq A A' \rightarrow A \cap B \subseteq A' \cap B :=$  by
2   assume A : set
3   assume B : set
4   assume A' : set
5   suppose A  $\subseteq$  A' that is equivalent to  $\forall Z, Z \in A \rightarrow Z \in A'$  as H1
6   we need to prove  $A \cap B \subseteq A' \cap B$  that is equivalent to  $\forall Z, Z \in A \cap B \rightarrow Z \in A' \cap B$ 
7   assume Z : set
8   suppose Z  $\in$  A  $\cap$  B
9   thus by ax_intersect1 we proved Z  $\in$  A as H2 and Z  $\in$  B as H3
10  we need to prove Z  $\in$  A'  $\cap$  B
11  by ax_intersect2 it suffices to prove Z  $\in$  A'  $\wedge$  Z  $\in$  B
12  by H1, H2 we proved Z  $\in$  A'
13  thus by H3 done
```

4.8 Teorema della monotonia dell'unione

Per ogni insieme X, X' e Y vale che:

$$X \subseteq X' \Rightarrow X \cup Y \subseteq X' \cup Y$$

Interpretazione: Se X è contenuto in X' , allora tutti gli elementi di X appartengono anche a X' . Di conseguenza, unendo X con un altro insieme Y , non si ottiene nulla che non sarebbe già presente nell'unione di X' con Y . In altre parole, l'unione è **monotona** rispetto all'inclusione: ampliando uno degli insiemi, l'unione può solo crescere o restare uguale, mai diminuire.

Dimostrazione

Siano X, X' , e Y insiemi tali che $X \subseteq X'$, ovvero $\forall Z, (Z \in X \Rightarrow Z \in X')$ (ipotesi H_1).
Dobbiamo dimostrare $X \cup Y \subseteq X' \cup Y$, ovvero $\forall Z, (Z \in X \cup Y \Rightarrow Z \in X' \cup Y)$.
Sia Z un insieme tale che $Z \in X \cup Y$.
Per il **teorema dell'unione binaria** dimostriamo che $Z \in X \vee Z \in Y$ (ipotesi H_2).
Per il **teorema dell'unione binaria** ci riduciamo a dimostrare $Z \in X' \vee Z \in Y$.
Procediamo per casi su H_2 :
- Caso $Z \in X$ (ipotesi K):
Dimostriamo $Z \in X'$.
Per H_1, K ovvio.
- Caso $Z \in Y$ (ipotesi K):

Dimostriamo $Z \in Y$.

Per K ovvio.

Qed.

5 Elementi matematici, relazioni, funzioni

Le basi della matematica vengono implementate a partire dalla teoria degli insiemi.

5.1 Coppie ordinate

In un insieme l'ordine non conta. Ad esempio $\{1, 2\} = \{2, 1\}$. Una **coppia ordinata** è invece una coppia dove c'è un primo e un secondo elemento, un concetto diverso da quello di insieme. Due coppie sono uguali se e solo se lo sono rispettivamente sia il primo che il secondo elemento.

Le coppie ordinate vengono indicate con le parentesi angolari:

$$\langle 1, 2 \rangle \neq \langle 2, 1 \rangle \neq \{1, 2\}$$

Una coppia non deve essere pensata come contenente (nel senso di \in) i suoi elementi:

$$2 \notin \langle 1, 2 \rangle$$

Le coppie ordinate vengono così definite: dati X e Y viene chiamata **coppia ordinata** di prima componente X e seconda componente Y il seguente insieme:

$$\langle X, Y \rangle := \{X, \{X, Y\}\}$$

5.1.1 Teorema di caratterizzazione delle coppie

Il **teorema di caratterizzazione delle coppie** afferma che due coppie sono uguali se e solo se lo sono rispettivamente sia il primo che il secondo elemento:

$$\langle X, Y \rangle = \langle X', Y' \rangle \iff X = X' \wedge Y = Y'$$

Dimostrazione

omessa.

5.1.2 Corollario del teorema delle coppie

Il corollario dice che:

$$\langle X, Y \rangle \neq \langle Y, X \rangle \text{ a meno che } X = Y$$

In pratica, scambiare gli elementi di una coppia, la rende una coppia diversa, a meno che i due elementi non siano uguali.

5.2 Prodotto cartesiano di insiemi

L'esistenza del **prodotto cartesiano di insiemi come insieme** è garantito dal seguente teorema:

$$\forall A, \forall B, \exists C, \forall Z, (Z \in C \iff \exists a, \exists b, (a \in A \wedge b \in B \wedge Z = \langle a, b \rangle))$$

L'insieme C viene chiamato **prodotto cartesiano** di A e B e lo indichiamo come $A \times B$. Intuitivamente esso è l'**insieme di tutte le coppie** dove il primo termine è un elemento di A e il secondo termine un elemento di B .

Esempio

$$\{a, b\} \times \{1, 2\} = \{\langle a, 1 \rangle, \langle a, 2 \rangle, \langle b, 1 \rangle, \langle b, 2 \rangle\}$$

5.3 Relazioni

Una **relazione** fra due insiemi A e B è un qualunque sottoinsieme del prodotto cartesiano $A \times B$. Posso esprimerla elencando in un insieme tutte le coppie ordinate che ne fanno parte.

Se \mathcal{R} è una relazione, possiamo scrivere $a\mathcal{R}b$ se e solo se $\langle a, b \rangle \in \mathcal{R}$.

Esempio

La relazione \leq sull'insieme numerico $\{0, 1, 2\}$ è definita così:

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

e $0 \leq 2$ è solo una notazione per $\langle 0, 2 \rangle \in \leq$

5.3.1 Relazioni da e verso insiemi vuoti

Il teorema riguardante le relazioni da e verso insiemi vuoti dice che:

$$\mathcal{R} \subseteq A \times \emptyset \Rightarrow \mathcal{R} = \emptyset$$

o anche

$$\mathcal{R} \subseteq \emptyset \times A \Rightarrow \mathcal{R} = \emptyset$$

Ovvero \mathcal{R} è solo la **relazione vuota**.

Dimostrazione intuitiva: non posso formare coppie prendendo uno dei due elementi dall'insieme vuoto, perché tale insieme è vuoto. Quindi l'insieme di tali coppie sarà vuoto.

5.4 Funzioni

Una **funzione** di dominio A e codominio B è una qualunque relazione $f \subseteq A \times B$ tale che:

$$\forall X, (X \in A \Rightarrow \exists! Y, X f Y)$$

Ovvero è una relazione per la quale per ogni elemento del dominio c'è un **unico** elemento del codominio in relazione con esso.

Se f è una funzione possiamo usare la seguente notazione:

$$y = f(x)$$

per indicare $x f y$, ovvero $\langle x, y \rangle \in f$

Una funzione è quindi semplicemente un insieme (anche infinito) di coppie. Da notare che non c'è però nessuna garanzia che essa sia calcolabile (che esista un metodo per calcolarla).

5.4.1 Spazio di funzioni

Dati due insiemi A e B è possibile dimostrare che esiste l'insieme che contiene tutte e sole le funzioni da A a B . In particolare il teorema dice che:

$$\forall A, \forall B, \exists C, \forall f, (f \in C \iff f \text{ è una funzione di dominio } A \text{ e codominio } B)$$

L'insieme C viene indicato come B^A e rappresenta lo **spazio delle funzioni** da A a B .

Abbiamo due casi particolare per le funzioni da/verso insiemi vuoti:

$$B^\emptyset = \{\emptyset\}$$

$$\emptyset^A = \emptyset \text{ se } A \neq \emptyset$$

Infatti, in quanto ogni funzione da A verso B è una relazione fra A e B , se A o B sono vuoti, le uniche relazioni sono la **relazione vuota** (già dimostrato). Però la relazione vuota è una funzione solo se è il dominio ad essere vuoto perché altrimenti a un elemento del dominio dovrei associare uno e un solo elemento del codominio, ma questo non ne ha in quanto vuoto.

5.5 Proprietà delle relazioni

Notazioni utili

In seguito utilizzeremo le seguenti notazioni:

1.

$$\forall X \in A, P(X)$$

per indicare che per ogni X in A vale $P(X)$, ovvero che $\forall X, (X \in A \Rightarrow P(X))$.

2.

$$\exists X \in A, P(X)$$

per indicare che esiste un X in A tale che $P(X)$, ovvero che $\exists X, (X \in A \wedge P(X))$

3.

$$\forall X, Y \in A, P(X, Y)$$

per indicare che $\forall X \in A, \forall Y \in A, P(X, Y)$

4.

$$\exists X, Y \in A, P(X, Y)$$

per indicare che $\exists X \in A, \exists Y \in A, P(X, Y)$

Sia $\mathcal{R} \subseteq A \times A$. La relazione \mathcal{R} può godere di una o più delle seguenti proprietà:

- **Riflessiva**, se

$$\forall X \in A, X \mathcal{R} X$$

- **Simmetrica**, se

$$\forall X, Y \in A, (X \mathcal{R} Y \Rightarrow Y \mathcal{R} X)$$

- **Transitiva**, se

$$\forall X, Y, Z \in A, (X \mathcal{R} Y \wedge Y \mathcal{R} Z \Rightarrow X \mathcal{R} Z)$$

Esempi

- $=$ gode di tutte e tre le proprietà, infatti $X = X$ (p. riflessiva), se $X = Y$ allora $Y = X$ (p. simmetrica) e se $X = Y, Y = Z$ allora $X = Z$ (p. transitiva).
- $<$ sui numeri naturali è transitiva, ma non simmetrica e non riflessiva.
- \leq sui numeri naturali è riflessiva e transitiva, ma non simmetrica.
- \neq è simmetrica, ma non riflessiva e transitiva (ad esempio se $1 \neq 2 \neq 1$ non è vero che $1 \neq 1$).

5.6 Tipi di relazioni

Sia $\mathcal{R} \subseteq A \times A$:

- \mathcal{R} è di **ordine stretto** se e solo se \mathcal{R} è **transitiva** e **non riflessiva**. Una relazione di ordinamento non deve per forza essere intesa come una retta, ma piuttosto come un albero.

Esempi

- $=, \neq, \leq$ non sono relazioni di ordinamento strette.
- $<$ è una relazione di ordinamento stretta.
- “Essere antenato di” è un ordinamento stretto sulle persone.

- \mathcal{R} è di **ordine (lasco)** se e solo se \mathcal{R} è **transitiva, riflessiva e antisimmetrica**

Esempi

- $=, \leq, \subseteq$ sono relazioni di ordinamento.
- $|$ (“divide”) è una relazione di ordinamento sui numeri naturali.
- $\neq, <$ non sono relazioni di ordinamento.

- \mathcal{R} è di **equivalenza** se e solo se \mathcal{R} è **transitiva, riflessiva e simmetrica**. Una relazione di equivalenza assomiglia all’uguaglianza, ma si concentra solo su un certo aspetto: l’uguaglianza è estremamente stretta, infatti due oggetti sono uguali solo se sono lo stesso oggetto, mentre l’equivalenza si limita a verificare che una (o più) caratteristica di due oggetti sia uguale.

Esempi

- $=$ è una relazione di equivalenza.
- “*avere lo stesso cognome*”, “*essere dello stesso modello*” sono relazioni di equivalenza.
- $\leq, <, \neq$ non sono relazioni di equivalenza.

5.7 Classi di equivalenza

Le relazioni di equivalenza di solito si indicano con dei simboli che ricordano l'equivalenza (per esempio \equiv). Inoltre a partire da una relazione di equivalenza $\equiv \subseteq A \times A$, possiamo definire la **classe di equivalenza di $x \in A$ rispetto a \equiv** nel seguente modo:

$$[x]_{\equiv} \stackrel{\text{def}}{=} \{y \in A \mid y \equiv x\}$$

In pratica $[x]_{\equiv}$ conterrà tutti gli elementi (appartenenti ad A) equivalenti a x , ovvero tutti gli elementi che hanno quella proprietà (espressa tramite \equiv) che ha x .

Per ognuna delle possibili proprietà di un elemento x particolare posso ottenere una diversa classe di equivalenza.

Il **teorema delle classi di equivalenza** afferma che:

Sia $\equiv \subseteq A \times A$ una relazione di equivalenza. Per ogni $x, y \in A$ accade una di queste due cose: o $[x]_{\equiv} = [y]_{\equiv}$ (quando $x \equiv y$) oppure $[x]_{\equiv}$ e $[y]_{\equiv}$ sono insiemi disgiunti (senza elementi in comune) (quando $x \not\equiv y$).

In pratica o la classe di equivalenza è la stessa o le classi di equivalenza sono insiemi disgiunti.

Dimostrazione

Per la proprietà transitiva di \equiv , se $x \equiv y$ allora ogni $z \in [x]_{\equiv}$ è tale che $z \equiv x \equiv y$ e quindi $z \in [y]_{\equiv}$ e perciò $[x]_{\equiv} = [y]_{\equiv}$.

Inoltre, per le proprietà simmetrica e transitiva di \equiv , se $z \in [x]_{\equiv} \cap [y]_{\equiv}$ allora $x \equiv z \equiv y$ e perciò $[x]_{\equiv} = [y]_{\equiv}$. Quindi le due classi sono identiche o disgiunte.

Nota: x appartiene sempre alla classe di equivalenza di x (di una certa proprietà) per riflessività.

Le classi di equivalenza rappresentano un'astrazione degli elementi, considerando una sola proprietà, e sono perciò particolarmente utili per **passare da una relazione di equivalenza a una di uguaglianza**, infatti se $x \equiv y$, non è vero che $x = y$, ma è vero che $[x]_{\equiv} = [y]_{\equiv}$.

Tramite le classi di equivalenza si può fare la **partizione di un insieme**: dato un insieme è possibile dividerlo in varie parti che non hanno elementi in comune.

Esempio di classe di equivalenza

Se la relazione \equiv è definita come “*avere la stessa lettera iniziale*”, le varie classi di equivalenza sono ad esempio:

$$[\text{"albero"}]_{\equiv} = \{\text{"albero"}, \text{"alga"}, \text{"armadillo"}, \dots\}$$

$$[\text{"alga"}]_{\equiv} = \{\text{"albero"}, \text{"alga"}, \text{"armadillo"}, \dots\}$$

$$[\text{"banana"}]_{\equiv} = \{\text{"banana"}, \text{"borsetta"}, \text{"bullo"}, \dots\}$$

ed è vero che:

$$[\text{"albero"}]_{\equiv} = [\text{"alga"}]_{\equiv} = [\text{"armadillo"}]_{\equiv} = \dots$$

$$[\text{"albero"}]_{\equiv} \cap [\text{"banana"}]_{\equiv} = \emptyset$$

5.8 Insieme quoziente

Sia $\equiv \subseteq A \times A$ una relazione di equivalenza. Possiamo definire l'**insieme quoziente** di A rispetto a \equiv nel seguente modo:

$$A_{/\equiv} \stackrel{\text{def}}{=} \{[x]_{\equiv} \mid x \in A\}$$

In pratica l'insieme $A_{/\equiv}$ rappresenta l'insieme contenente tutte le classi di equivalenza degli elementi di A rispetto ad una certa relazione di equivalenza (\equiv). L'esistenza di tale insieme è garantita dall'assioma di rimpiazzamento.

Esempio

“*avere la stessa età*” è una relazione di equivalenza sulle persone. Se la indichiamo con \equiv , allora possiamo dire che ad esempio $[Andrea]_{\equiv}$ = tutte le persone che hanno 19 anni. E così via. Avremo quindi che $Persone_{/\equiv}$ (l’insieme quoziente delle persone rispetto all’avere la stessa età) conterrà un elemento per ogni possibile età, e tale elemento sarà l’insieme di tutte le persone che hanno la tale età.

Possiamo intuire il fatto che gli insiemi quozienti sono strumenti potenti che permettono di implementare nuovi concetti a partire da concetti pre-esistenti, per poi “nascondere” i dettagli dovuti alla rappresentazione.

5.8.1 Esempio: i numeri interi \mathbb{Z}

Poiché nei numeri naturali alcune operazioni, come la sottrazione, non sono chiuse, vorremmo costruire nuovi insiemi numerici che le permettano.

L’introduzione dei numeri interi permette di fare sottrazioni fra numeri naturali arbitrari (es. $4 - 2$).

Costruzione di \mathbb{Z} :

- Definiamo un insieme $Z = \mathbb{N} \times \mathbb{N}$ i cui elementi, che sono coppie ordinate $\langle n, m \rangle$, rappresenteranno intuitivamente i valori del tipo $n - m$.
- Ora abbiamo bisogno di una relazione di equivalenza che permetta di affermare che ad esempio $\langle 2, 4 \rangle$ sia equivalente a $\langle 3, 5 \rangle$ (infatti vogliamo che $2 - 4 = 3 - 5$).
- Definiamo la relazione $\equiv \subseteq Z \times Z$:

$$\langle u_1, l_1 \rangle \equiv \langle u_2, l_2 \rangle \stackrel{\text{def}}{=} u_1 + l_2 = u_2 + l_1$$

In pratica passiamo dalla sottrazione all’addizione, in quanto questa è definita in \mathbb{N} , e così possiamo affermare che $\langle 2, 4 \rangle \equiv \langle 3, 5 \rangle$ perché rappresentano la stessa sottrazione, poiché $2 + 5 = 4 + 3$ (infatti $2 - 4 = 3 - 5 \iff 2 + 5 = 4 + 3$).

- È facile dimostrare che \equiv è una relazione di equivalenza.
- Ora possiamo quindi definire l’insieme dei numeri interi come l’insieme quoziente di Z rispetto a \equiv :

$$\mathbb{Z} \stackrel{\text{def}}{=} Z_{/\equiv}$$

$$\mathbb{Z} = \{ \dots, [\langle 0, 2 \rangle]_{\equiv}, [\langle 0, 1 \rangle]_{\equiv}, [\langle 0, 0 \rangle]_{\equiv}, [\langle 1, 0 \rangle]_{\equiv}, [\langle 2, 0 \rangle]_{\equiv}, \dots \}$$

- Indicheremo $[\langle 0, i \rangle]_{\equiv}$ con $-i$, $[\langle 0, 0 \rangle]_{\equiv}$ con 0 e $[\langle i, 0 \rangle]_{\equiv}$ con $+i$, perciò avremo che:

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

A seguito di questa costruzione possiamo intendere il fatto che il “2” numero naturale è diverso dal “2” numero intero in quanto hanno rappresentazioni (costruzioni) diverse. Il matematico però di solito fa implicitamente il passaggio tra uno e l’altro.

5.9 Proprietà delle funzioni

Sia $f \in B^A$ una funzione di dominio A e codominio B . La funzione f può essere:

1. iniettiva se

$$\forall x, y \in A, (f(x) = f(y) \Rightarrow x = y)$$

ovvero se non esistono due elementi distinti del dominio che mappano allo stesso elemento nel codominio (possono però esserci punti addizionali del codominio che non utilizzo).

2. suriettiva se

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

ovvero se tutti i punti del codominio sono mappati a un elemento del dominio (magari anche a più di uno).

3. biettiva se

f è sia iniettiva che suriettiva

Esempi

- $+1$ è biettiva sui numeri interi, iniettiva ma non suriettiva sui naturali (0 nel codominio non viene mappato con nessun elemento del dominio, ovvero non c'è nessun numero naturale n tale che $n + 1 = 0$).
- $|z|$ (il valore assoluto) è non suriettiva e non iniettiva sugli interi.

A questo punto possiamo iniziare a ragionare sul concetto di dimensione di un insieme e possiamo provare a formulare le seguenti intuizioni:

- Se f è iniettiva allora B ha **almeno** tanti elementi quanti ne ha A .
- Se f è suriettiva allora A ha **almeno** tanti elementi quanti ne ha B .
- Se f è biettiva allora A e B hanno lo stesso “numero” (la stessa quantità) di elementi.

Questa intuizione è buona, ma non possiamo ragionare in termini di numeri: quanti elementi (in numero) ha un insieme infinito? Ci sono insiemi più infiniti di altri?

Per rispondere a queste domande dovremo approfondire il concetto di cardinalità/taglia di un insieme.

5.10 Cardinalità

Prima di introdurre la definizione generale di cardinalità possiamo intanto affermare che: due insiemi A, B hanno la stessa cardinalità se e solo se esiste una biiezione fra A e B .

Notiamo che “avere la stessa cardinalità” è una relazione di equivalenza, ma sulla classe di tutti gli insiemi (che è appunto una classe, non un insieme). Nel procedere con le definizioni di numeri cardinali e cardinalità utilizzeremo perciò le classi. Sarebbe possibile fare lo stesso anche partendo da insiemi, ma sarebbe molto più complesso.

5.10.1 Numeri cardinali

Sia U la classe di tutti gli insiemi. Un **numero cardinale** è un elemento di $U_{/\equiv}$ dove \equiv è la relazione di equivalenza “avere la stessa cardinalità”.

Se l'insieme preso in considerazione è un insieme finito, il numero cardinale corrispondente utilizzerà la stessa notazione del numeri naturale che indica il numero di elementi di tale insieme.

Se invece l'insieme è infinito il numero cardinale corrispondente (ovvero la classe di equivalenza di tale insieme) viene indicato con le lettere dell'alfabeto ebraico, come aleph (\aleph): la cardinalità di \mathbb{N} ad esempio è indicata con \aleph_0 .

Nota: la classe di equivalenza per l'insieme vuoto contiene solo il vuoto.

Esempi

$$3 \stackrel{\text{def}}{=} [\{ 1, 2, 3 \}]_{\equiv} = \{ \{ 1, 2, 3 \}, \{ 5, 2, 8 \}, \{ \emptyset, 3, \langle 1, 2 \rangle \}, \dots \}$$

$$0 \stackrel{\text{def}}{=} [\emptyset]_{\equiv} = \{ \emptyset \}$$

$$\aleph_0 \stackrel{\text{def}}{=} [\mathbb{N}]_{\equiv} = \{ \{ n \in \mathbb{N} \mid n \text{ è pari} \}, \mathbb{Z}, \mathbb{Q}, \dots \}$$

Come possiamo vedere, il fatto che un insieme infinito sia un sottoinsieme di un altro **non vuol dire che essi abbiano cardinalità diverse**. Ad esempio \mathbb{N} è sottoinsieme di \mathbb{Z} , ma in realtà hanno la stessa cardinalità (\aleph_0).

5.10.2 Insiemi enumerabili

Gli insiemi che hanno come cardinalità \aleph_0 , sono detti **enumerabili**.

È possibile dimostrare che \mathbb{Z} e \mathbb{Q} sono numerabili, mentre invece \mathbb{R} non lo è. Quest'ultima dimostrazione è stata formulata da Cantor, utilizzando il metodo di diagonalizzazione di Cantor.

Si può addirittura arrivare a dimostrare l'esistenza di infinite diverse cardinalità di infiniti, e che l'insieme di queste è un infinito non enumerabile.

Nota: tutte le funzioni che si possono scrivere in un linguaggio di programmazione sono enumerabili. Come conseguenza di ciò in informatica non è possibile implementare tutte le possibili funzioni.

5.10.3 Definizione di cardinalità

Dato un insieme A , si definisce **cardinalità** di A il **numero cardinale** $|A|_{\equiv}$ e la si indica con la seguente notazione:

$$|A|$$

Esempio

$$|\{1, 2, 3\}| = |\{\{1, 2, 3\}\}|_{\equiv} = \{\{1, 2, 3\}, \{5, 2, 8\}, \{\emptyset, 3, \langle 1, 2 \rangle\}, \dots\} = 3$$

5.10.4 Ordinamento dei numeri cardinali

In quanto utilizziamo i numeri cardinali per definire la taglia degli insiemi, vogliamo definire una relazione di ordinamento tra di essi, in modo da poter confrontare le diverse cardinalità.

Siano x, y due numeri cardinali. Diremo che $x \leq y$ se, dati due insiemi A e B tali che $|A| = x$ e $|B| = y$, **esiste una iniezione tra A e B** . In particolare diremo che $|A| \leq |B|$ se e solo se esiste una iniezione tra A e B .

Siano x, y due numeri cardinali. Diremo che $x < y$ se, $x \leq y$ e $x \neq y$. In particolare diremo che $|A| < |B|$ se e solo se esiste una iniezione e nessuna biiezione tra A e B .

Esempi

- $2 = |\{1, 2\}| < |\{a, b, c\}| = 3$ come testimoniato dall'iniezione $1 \mapsto a, 2 \mapsto b$ e dall'assenza di biezioni.
- $2 = |\{1, 2\}| < |\mathbb{N}| = \aleph_0$ come testimoniato dall'iniezione $1 \mapsto 1, 2 \mapsto 2$ e dall'assenza di biezioni.
- $|\mathbb{P}| = |\{n \in \mathbb{N} \mid n \text{ è pari}\}| \leq |\mathbb{N}|$ come testimoniato dalla funzione identità che è una iniezione.
- $|\mathbb{P}| \not< |\mathbb{N}|$ in quanto $|\mathbb{P}| = |\mathbb{N}|$ come testimoniato dalla biezione $f(x) = 2 \cdot x, f \in \mathbb{N}^{\mathbb{P}}$

5.11 Insiemi finiti e infiniti

Anche se intuitivamente possiamo intendere le differenze tra insiemi finiti e insiemi infiniti, cerchiamo di dare definizioni più rigorose.

Innanzitutto chiameremo **finito** un insieme che **non è infinito**.

Un insieme si dice invece **infinito** quando è **in biiezione con un suo sottoinsieme proprio**. Ovvero se A è un insieme e B un suo sottoinsieme proprio ($B \subsetneq A$), se $|B| = |A|$ allora A è infinito.

5.12 Teorema di Cantor

L'enunciato del teorema di Cantor è il seguente:

sia T un insieme non vuoto. Allora la sua cardinalità è strettamente minore della cardinalità dell'**insieme delle parti di T** , ovvero $|T| < |2^T|$.

Il corollario del teorema di Cantor invece dice che:

sia T è un insieme con almeno due elementi. Allora la cardinalità di T è strettamente minore della cardinalità dello **spazio delle funzioni di T** , ovvero $|T| < |T^T|$.

6 Semantica classica

Quando si studia la logica, ortogonalmente (indipendentemente) si scelgono una sintassi (che regole logiche) e una semantica (che significato hanno le parole). La semantica è proprio quella parte della linguistica che studia il **significato delle parole**.

Come semantica quasi la totalità dei matematici (della matematica) usano la **semantica classica**.

6.1 Terminologia

- **Connotazione**: una parte di frase (presa da un insieme di possibili frasi) sintatticamente corretta, alla quale attribuisco un significato.
- **Denotazione**: il significato attribuito a una connotazione, preso da un dominio di interpretazione.
- **Dominio di interpretazione**: insieme di possibili significati.
- **Sintassi**: descrizione dell'insieme di tutte le connotazioni.
- **Interpretazione / (funzione) semantica**: funzione che associa a ogni connotazione una denotazione in un dominio di interpretazione fissato

In caso di ragionamento ipotetico, ogni possibile **mondo** (= configurazione: interpretazioni diverse per la stessa connotazione) ha una semantica associata.

Si possono quindi dare semantiche totalmente diverse allo stesso linguaggio.

In genere c'è una semantica "naturale" o "principale", detta **semantica intesa** (quella che si intende se non si precisa nulla, qualcosa su cui ci si è accordati). Le altre semantiche sono dette **alternative**.

6.2 Semantica classica della logica proposizionale

La **logica proposizionale** è la logica, già vista in precedenza, in cui ogni connotazione può denotare un valore di verità.

Come già detto però a una logica va associata anche una semantica. Vediamo come la semantica classica si applica alla logica proposizionale:

- A ogni denotazione viene associato il suo **valore di verità** in un qualche mondo: il valore di verità non è assoluto, ma relativo a un mondo (fissato in ogni mondo, ma da mondo a mondo può cambiare).
- Il mondo determina il valore di verità delle variabili proposizionali A, B, \dots . Ogni mondo avrà quindi i propri valori di verità per ogni possibile variabile.
- La semantica dei connettivi ($\wedge, \Rightarrow, \dots$) è invece **fissata**, non determinata dal mondo.
- Quando si assume la semantica classica, la logica si dice **logica proposizionale classica**.

6.3 Logica classica

Fissata la semantica (classica) bisogna però anche capire cosa vuol dire essere vero, **cos'è la verità**.

La **logica classica** segue la visione Platonica del mondo: esiste un mondo delle idee perfetto e queste regole di verità descrivono tale mondo perfetto. In particolare per ogni mondo (preso singolarmente) vale che:

1. Ogni enunciato è **vero o falso**: siamo in un piano deterministico e non probabilistico, non sono ammesse sfumature di significato (valori di verità diversi), ma solo due distinti valori di verità.
2. Un enunciato non può essere vero e falso allo stesso tempo: **principio di non contraddizione**. Questo vale anche nelle logiche probabilistiche, dove la "somma" dei valori di verità deve essere sempre 1. Nelle logiche che invece permettono la contraddizione, il ragionamento logico diventa argomentazione (teoria dell'argomentazione).
3. Il valore di verità non muta: **principio di staticità**. Visione deterministica e statica del mondo.
4. Il valore di verità di un enunciato è sempre determinato: **principio di determinatezza**. Ad esempio per la matematica questo principio dipende dalla visione che il matematico segue:
 - la matematica è già tutta esistente: il matematico la "scopre"
 - inizialmente non esistono dei concetti, sono indeterminati: il matematico la "crea"

Analizziamo i punti:

- 1 e 2 determinano il **dominio di interpretazione classico**: $\{0, 1\}$. Gli elementi di tale insieme rappresentano le notazioni classiche di verità e falsità: si potrebbe scegliere un qualunque insieme con due elementi, ma vengono utilizzati 0 e 1 per 3 motivi:
 - si evita di fare errori tarskiani (ad empio ambiguità con le parole vero e falso).
 - sui numeri ho delle operazioni matematiche che è possibile usare per dare significato ai connettivi logici.
 - questa scelta scala a logiche diverse: facile conversione ad esempio l'intervallo aperto $[0, 1]$ nelle logiche probabilistiche.
- 3 e 4 determinano che un dominio **non muta**: ogni formula logica deve aver associato un valore di verità che non cambia. Questo viene rappresentato con una funzione (matematica): **una (funzione di) interpretazione (classica) o mondo** è una funzione dall'insieme delle variabili proposizionali $\{A, B, \dots\}$ verso $\{0, 1\}$

Al di fuori della logica classica i punti 3 e 4 sono spesso falsi (es. in logica intuizionista), in quando parlano di staticità e immutabilità della verità.

Indichiamo le interpretazioni (mondi) con v, v', v_1, v_2, \dots

Cambiare mondo vuol dire cambiare la funzione che associa alle variabili i valori 0 e 1.

Essendo la sintassi definita in modo ricorsivo, una formula sintatticamente corretta non è semplicemente una stringa, ma un albero. L'albero associato a una formula si chiama **albero di sintassi astratta**. Ogni sotto-albero corrisponde a una sotto-formula.

Per calcolare il valore di verità di una formula si procede per ricorsione: parto dalle foglie ($A, B, C \dots$) di cui so già i valori di verità, poi salgo componendo i risultati precedenti (prima calcolo il sotto-albero di sinistra, poi quello di destra, poi combino i risultati). Questo processo permette, grazie alla **ricorsione strutturale**, di calcolare il valore di verità espresso da un albero.

Per ricorsione strutturale possiamo anche definire il valore di verità di una formula (in logica classica). Usiamo le doppie parentesi quadre $[[\cdot]]$ per passare dalla sintassi alla semantica (es. "La semantica di un certo mondo v è definita nel seguente modo...").

Data un'interpretazione (o mondo) v , la semantica $[[\cdot]]^v : F \rightarrow \{0, 1\}$, è definita per ricorsione strutturale sulle connotazioni come segue:

$$\begin{aligned}
 [[\perp]]^v &= 0 \\
 [[\top]]^v &= 1 \\
 [[A]]^v &= v(A) \\
 [[\neg F]]^v &= 1 - [[F]]^v \\
 [[F_1 \wedge F_2]]^v &= \min\{[[F_1]]^v, [[F_2]]^v\} \\
 [[F_1 \vee F_2]]^v &= \max\{[[F_1]]^v, [[F_2]]^v\} \\
 [[F_1 \Rightarrow F_2]]^v &= \max\{1 - [[F_1]]^v, [[F_2]]^v\}
 \end{aligned}$$

Un mondo v è una funzione che associa a ogni parametro un valore 0 o 1, quindi $v(A)$ vuol dire il valore di A nel mondo v .

6.4 Conseguenza logica

Γ rappresenta un insieme di affermazioni, ipotesi.

Definizione: $\Gamma \models F$ (F è **conseguenza logica** di Γ) quando per ogni mondo v si ha che, se $[[G]]^v = 1$ per ogni $G \in \Gamma$, allora $[[F]]^v = 1$

ovvero quando:

- Se il minimo dell'insieme dei valori di verità (delle formule in Γ) vale 1, allora il valore di verità di F deve valere 1.

- F vale in tutti quei mondi in cui tutte le formule di Γ valgono.
- L'insieme dei mondi in cui tutte le formule di Γ valgono è un sottoinsieme di quello in cui vale F : ogni ipotesi è un filtro, riduce il campo di interesse.
- Se l'intersezione di tali mondi è vuota, le ipotesi sono inconsistenti ed essendo che il vuoto è sottoinsieme di ogni cosa, F vale. Infatti ogni cosa è conseguenza logica di un insieme di ipotesi inconsistenti.
- Più formule (ipotesi) metto più aumentano le conseguenze logiche, ma queste si applicano in sempre meno casi. Ottengo situazioni più specifiche, ma posso dimostrare più cose.

7 Semantica intuizionista

7.1 Introduzione

La semantica non della verità platonica, ma dell'evidenza (qualcosa che mi convince).

La semantica della conoscenza diretta del fenomeno.

La semantica della calcolabilità (posso risolvere un problema con un programma): semantica degli informatici

Anche la visione della matematica cambia a seconda della logica usata:

- la matematica è una scoperta per la logica classica
- la matematica è un'invenzione per la logica intuizionista

Nella semantica classica sapere che c'è un risultato (la verità) non implica sapere cos'è (conoscenza della verità): si parla quindi di **evidenza indiretta**.

Nella semantica intuizionista è invece importante avere un'**evidenza diretta**.

Facciamo un esempio con la formula $\exists x.P(x)$:

- in logica classica non interessa sapere qual'è quel x tale che $P(x)$, basta sapere che esiste.
- in logica intuizionista si vuole invece avere un algoritmo che permetta di trovare tale x .

Per questo in tutte le dimostrazioni intuizioniste del tipo: "per ogni input esiste un output in relazione con l'input", è presente un algoritmo che calcola l'output a partire dall'input.

Una dimostrazione classica invece dice solo che l'output esiste, non spiega come calcolarlo.

Come già visto, nella semantica classica:

- Il valore di verità di ogni enunciato è sempre determinato
- Il valore di verità di ogni enunciato è immutabile

Queste ipotesi descrivono però un mondo platonico e perfetto, ma non sono appropriate per la descrizione della conoscenza e per mondi non deterministici.

Perciò, nelle semantiche intuizioniste:

- Il valore di verità di ogni enunciato è determinato solo quando se ne ha una prova/evidenza **diretta** (un **algoritmo**).
- Il valore di verità di ogni enunciato può passare in maniera monotona dall'essere indeterminato all'avere un determinato valore che non cambia più (scopro almeno un algoritmo o dimostro che non può esserci)

Ci sono però problemi per i quali un algoritmo può dimostrare solo uno dei due possibili stati di verità: nell'altro caso non terminerebbero mai

7.2 Semantiche

Esistono due principali semantiche per la logica intuizionista:

- **Semantica alla Kripke (o dei mondi possibili):**
 - Continua a usare due valori (l'insieme $\{0, 1\}$), ma con significato diverso: 1 = so che è vero, 0 = indeterminato. Una formula A è falsa quando $\neg A$ vale 1, non quando A vale 0.
 - I mondi continuano ad essere delle funzioni che assegnano alle variabili una denotazione 0 o 1. Ma a differenza della logica classica i mondi possono evolvere: posso passare da un mondo in cui $v(A) = 0$ a un mondo v' uguale a v tranne per il fatto che $v'(A) = 1$.
 - Tanti modi possibili in evoluzione possibile (l'evoluzione non è dovuta al tempo, ma alla conoscenza).
 - Essendo che il valore 0 non indica il falso, ma la non conoscenza, non è vero il principio del terzo escluso: $v \Vdash A \vee \neg A$, se $v(A) = 0$
 - Le denotazioni $\{0, 1\}$ sono livelli di conoscenza, non algoritmi.
- **Semantica di Brouwer-Heyting-Kolmogorov**
 - Una formula F è la descrizione di un problema, di un programma.
 - A ogni problema F associo l'**insieme di evidenze**, ovvero l'**insieme di algoritmi conosciuti** per il problema F .
 - Insieme vuoto = assenza di algoritmi \approx falsità
 - Insieme non vuoto = almeno un algoritmo \approx verità

- Un connettivo compone problemi più complessi a partire da problemi più semplici
- La denotazione di un connettivo è un insieme di algoritmi che risolvono il problema composto usando gli algoritmi per i problemi semplici

Useremo la semantica di Brouwer-Heyting-Kolmogorov

Definizioni:

$$[[\perp]]^v = \emptyset$$

nessun algoritmo per risolvere \perp

$$[[\top]]^v = \{*\}$$

\top e' un problema banale che $*$ puo' risolvere

$$[[A]]^v = v(A)$$

l'insieme di algoritmi che risolvono A

$$[[F \wedge G]]^v = [[F]]^v \times [[G]]^v$$

prodotto cartesiano: coppie di un algoritmo per F e uno per G

$$[[F \vee G]]^v = [[F]]^v \oplus [[G]]^v$$

unione disgiunta: risolvere uno dei due problemi, dicendo quale

coppie con un booleano e un algoritmo (0 = algoritmo per F , 1 = algoritmo per G)

$$[[F \Rightarrow G]]^v = [[G]]^{[[F]]^v}$$

trovare un algoritmo per risolvere G , sapendo l'algoritmo risolutivo di F

7.3 Decidibilità, correttezza e completezza

Una proposizione è **decidibile** se esiste un algoritmo che ci dice se essa è vera o falsa. In generale però in logica intuizionista $\not\models A \vee \neg A$, ovvero non vale l'excluded middle.

Teorema: per ogni Γ, F , se $\Gamma \vdash F$ in logica intuizionista allora $\Gamma \Vdash F$ anche in logica classica (non è però sempre vero il contrario, in quanto la logica classica ci parla solo dell'esistenza della verità, non di come trovarla)

Teorema (correttezza): per ogni Γ, F , se $\Gamma \vdash F$ senza usare il ragionamento per assurdo (RAA) allora $\Gamma \Vdash F$ in logica proposizionale intuizionista.

Teorema (completezza debole): per ogni insieme finito di formule Γ e per ogni F , se $\Gamma \Vdash F$ in logica proposizionale intuizionista allora $\Gamma \vdash F$ senza usare la RAA.

8 Deduzione naturale

Per studiare le **prove** bisogna prima introdurre una definizione rigorosa. Una possibilità della definizione di prova sono gli **alberi di deduzione naturale**, che sono un **dato** che rappresenta una prova.

8.1 Logica proposizionale

La **logica proposizionale** è quella logica, meno ricca di quella vista finora, che studia solo alcune formule logiche. La studiamo non per fare teoremi matematici ma perché ha anche tante applicazioni in informatica.

Non tutto ciò che si può scrivere è una **proposizione** (es. “2” non è una proposizione): una proposizione è qualcosa che **vale** o **non vale** (qualunque sia il significato di valere).

Sintassi della logica:

- \perp (“bottom” / “falso” / “assurdo”): rappresenta ciò che **non vale mai**
- \top (“top” / “vero”): rappresenta la verità, che **vale sempre**.
- A, B, \dots sono **variabili proposizionali** che rappresentano qualcosa che **può o meno valere**. La loro “risposta” non è univoca, ma dipende dalle ipotesi
- $\neg F_1$ (“non”): **negazione** di F_1
- $F_1 \wedge F_2$ (“e”): **congiunzione** di F_1 e F_2
- $F_1 \vee F_2$ (“o”): **disgiunzione (non esclusiva)** di F_1 e F_2
- $F_1 \Rightarrow F_2$ (“se...allora”): **implicazione** fra F_1 e F_2

Nota

Il concetto di “**valere**” non è ancora definito.

\neg è un **connettivo unario**, $\vee, \wedge, \Rightarrow$ sono **connettivi binari**, mentre \top, \perp sono **connettivi 0-ari** o anche detti **costanti**.

Per convenzione i connettivi binari sono **associativi a destra**. Inoltre \neg ha la precedenza su \wedge che ha precedenza su \vee che ha precedenza su \Rightarrow .

Esempio

$$A \wedge B \Rightarrow \neg C \vee A \quad \text{significa} \quad (A \wedge B) \Rightarrow ((\neg C) \vee A)$$

Già così la logica proposizionale permette di catturare molti ragionamenti validi. Più avanti però introdurremo logiche più ricche, più complete che fanno uso anche di **quantificatori, predicati, funzioni, costanti e variabili**.

Formalizzare una proposizione/frase/testo significa tradurla in **formule logiche**. Il processo di formalizzazione però non sempre è possibile o ovvio e spesso è approssimato. Inoltre un enunciato formalizzato cattura un'intera famiglia di ragionamenti informali con la stessa struttura logica, ovvero esso è un ragionamento universalmente valido in tali istanze.

Esempio

Consideriamo la seguente frase:

“Se oggi piove allora prendo l'ombrello, se prendo l'ombrello non mi bagno. Quindi se oggi piove allora non mi bagno” Possiamo formalizzarla nel seguente **enunciato** (che è dimostrabile):

$$(P \Rightarrow O) \wedge (O \Rightarrow \neg B) \Rightarrow P \Rightarrow \neg B$$

Notiamo però che l'enunciato così posto non si riferisce unicamente alla frase di partenza, ma all'intera famiglia di ragionamenti con quella struttura. Ad esempio anche la seguente frase può essere rappresentato da tale enunciato:

“Se x è un multiplo di 4 allora 4 divide x , se 4 divide x allora x non è dispari. Quindi se x è un multiplo di 4 allora x non è dispari”

8.2 Alberi di deduzione naturale

In logica, come in informatica, un **albero** è una **struttura dati** composta da **nodi**, collegati tra loro tramite **archi**, a partire da un nodo iniziale detto **radice**. I nodi che non presentano ulteriori ramificazioni sono detti **foglie**.

Un altro modo di interpretare la struttura ad albero consiste nel considerare le ramificazioni non semplicemente come collegamenti a nodi successivi, ma come connessioni a **sotto-alberi**. Ogni sotto-albero possiede a sua volta una radice e, eventualmente, ulteriori sotto-alberi discendenti. Le radici dei sotto-alberi che sono privi di ulteriori sotto-alberi coincidono pertanto con le foglie. Ne segue che un albero può essere definito **ricorsivamente** come:

- una foglia (equivalente a una radice), oppure
- un nodo che si ramifica in uno o più sotto-alberi.

Per questo motivo, la struttura degli alberi è detta **ricorsiva**.

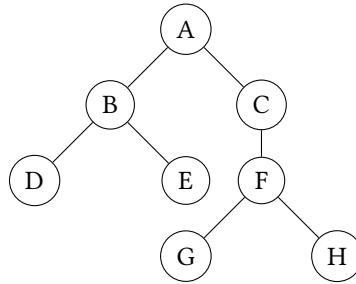


Figure 1: Esempio di albero in informatica con radice A e ramificazioni in sotto-alberi.

La stessa idea ricorsiva può essere applicata alle **dimostrazioni**: a ogni passaggio ci si trova in una nuova sotto-dimostrazione, che può eventualmente suddividersi in ulteriori sotto-dimostrazioni, e così via. Per lavorare con questa rappresentazione delle dimostrazioni utilizzeremo perciò quelli che vengono chiamati **alberi di deduzione naturale**.

Gli **alberi di deduzione naturale** sono strutture dati analoghe a quelle utilizzate in informatica. Tuttavia, in analogia con le rappresentazioni botaniche, si tende a rappresentare la radice (**top**) nella parte inferiore e le foglie (**bottom**) nella parte superiore dell'albero. Sono dunque presenti ramificazioni e foglie, tutte considerate come nodi; le foglie si distinguono in quanto nodi privi di ulteriori ramificazioni.

A ogni nodo viene associata un'informazione (una formula), e lo stesso vale per le ramificazioni. La costruzione di un albero di deduzione parte da una radice, corrispondente a una formula F . Le ramificazioni sono rappresentate tramite linee orizzontali: sopra ciascuna di esse compaiono zero o più altre formule.

Procedendo ricorsivamente da queste formule, si generano ulteriori ramificazioni e nodi; alcune di queste formule costituiranno eventualmente foglie. Una linea orizzontale che non presenta formule superiori indica una ramificazione nel "nulla": questa situazione non corrisponde a una foglia.

$$\frac{\frac{[A \wedge (B \Rightarrow C)]}{B \Rightarrow C} \wedge e_2 \quad B \Rightarrow e}{\frac{C}{A \wedge (B \Rightarrow C) \Rightarrow C} \Rightarrow i}$$

Figure 2: Esempio di albero di deduzione naturale.

Un albero di deduzione naturale viene utilizzato per dimostrare una proposizione del seguente tipo:

$$\Gamma \vdash F$$

dove Γ è un insieme di formule (I_1, I_2, \dots, I_n) , dette **ipotesi**, \vdash è il simbolo con il significato di "**deriva**" (si legge "Γ deriva F" o "F derivato da Γ") e F è una formula che rappresenta la conclusione da dimostrare.

Un albero per essere una dimostrazione di $\Gamma \vdash F$ deve avere le seguenti proprietà:

- Tutti i nodi sono etichettati con delle formule
- Le foglie possono essere di due tipi:

- **Formule scaricate** (cancellate). Vengono indicate tra parentesi quadre e rappresentano **ipotesi locali** di una sotto-dimostrazione
- Formule non scaricate, in tal caso rappresentano **ipotesi globali**
- La radice deve essere ciò che si deve dimostrare (F)
- Le foglie non scartate (globali) devono essere un **sottoinsieme delle ipotesi di partenza** (γ)
- Le ramificazioni hanno delle etichette che rappresentano il nome della **regola di inferenza** utilizzata.

8.3 Sintassi per le regole di inferenza

Per le regole di inferenza viene usata la seguente sintassi:

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

La formula F rappresenta la **conclusione** della regola.

Le formule F_1, \dots, F_n sono le **premesse** della regola.

A fianco della riga viene indicato il nome della regola.

Inoltre con la sintassi

$$\begin{array}{c} [A] \\ \vdots \\ F_i \end{array}$$

viene indicato che è possibile assumere localmente A per concludere la premessa F_i . A sarà quindi un'ipotesi scaricata e potrà essere usata solo in quel ramo della dimostrazione.

Una regola senza ipotesi ($n = 0$) viene chiamata **assioma**, rappresenta una regola assoluta.

Gli alberi di deduzione naturali si compongono ricorsivamente dalle **regole di inferenza**, creando delle sotto-dimostrazioni.

La struttura ricorsiva permette di definire **funzioni per ricorsione strutturale** su alberi di deduzione e di effettuare **prove per induzione strutturale** (per dimostrare che la dimostrazione è corretta).

Ci sono due tipi di passi di inferenza:

- **Regole di introduzione:** determinano tutti i modi in cui è possibile concludere una formula con in testa un determinato connettivo. “Come concludo...?”
- **Regole di eliminazione:** determinano i modi in cui è possibile utilizzare un'ipotesi con in testa un determinato connettivo. “Cosa ricavo da...?”

Ogni singola regola (e di conseguenza ogni albero), ammette sempre due diverse letture:

- **Top-down:** dalla conclusione alle premesse (negli alberi quindi dalla radice a salire verso le foglie). “Per concludere F posso ridurmi a dimostrare $F_1 \dots F_n$ ”
- **Bottom-up:** dalle premesse alla conclusione. “Date le premesse $F_1 \dots F_n$ posso concludere F ”

Spesso nel leggere un albero (una dimostrazione) si usa una lettura mista: delle parti top-down, delle parti bottom-up.

I matematici di solito lavorano in modo top down (partono da ciò che devono dimostrare), infatti è più facile, poiché avendo una sola conclusione le possibilità spesso sono limitate. Però poi quando il matematico presenta la dimostrazione, lo fa in modo bottom up

8.4 Correttezza delle regole di inferenza

Una regola $\frac{F_1 \dots F_n}{H}$ (*nome*) è **corretta** quando $F_1, \dots, F_n \Vdash H$.

Il simbolo \Vdash rappresenta una **conseguenza logica** ed è diverso da \vdash . Una forma del tipo $\Gamma \Vdash F$ si legge come “ F è una conseguenza logica di Γ ” e significa che “in tutti quei casi in cui valgono tutte le formule in Γ si ha che F vale”. In quanto il significato di “valere” cambia da logica a logica, il significato della conseguenza logica cambia

da logica a logica.

\Vdash non rappresenta una dimostrazione, ma un fatto naturale, sempre vero.

Inoltre se una premessa contempla ipotesi scaricate, esse vanno integrate tramite **implicazioni** nella formula finale.

Esempio

$$\frac{E \quad \begin{array}{c} [F] \\ \vdots \\ G \end{array}}{H}$$

Questa formula è corretta quando $E, F \Rightarrow G \Vdash H$

Quindi una regola corretta **dimostra solo conseguenze logiche**.

8.5 Invertibilità delle regole

Una regola $\frac{F_1 \dots F_n}{F}$ è **invertibile** quando per ogni i si ha che F ha come **conseguenza logica** F_i ($F \Vdash F_i$). Inoltre come per la completezza, eventuali ipotesi scaricate (p.e. H) di F_i vanno integrate con una implicazione (es. $F \Vdash H \Rightarrow F_i$).

In pratica l'invertibilità ci dice che **se vale la conclusione valgono anche le premesse**.

L'invertibilità è importante nella ricerca delle prove, in quanto una regola invertibile può essere applicata nella ricerca top-down in maniera sicura, in quanto sarà sempre la scelta corretta. È quindi importante ricordarsi quali regole sono invertibili e quali no.

Ciò non vuol dire che una regola non invertibile non sia corretta, ma che, se applicata al momento sbagliato, potrebbe portare a una formula impossibile da dimostrare.

Per questo motivo, se ci si trova in un vicolo cieco durante una dimostrazione, bisogna fare backtracking e provare ad applicare un'altra regola, ma solo per le regole non invertibili, in quanto quelle invertibili sappiamo essere corrette sempre.

In alcuni casi certe premesse potrebbero essere conseguenza logica di F , mentre altre no: in questi casi la regola si dice **parzialmente invertibile**.

8.6 Derivabilità delle regole

Un insieme di regole \mathcal{R} è **derivabile** a partire da un insieme di regole \mathcal{S} quando per ogni regola $\frac{F_1 \dots F_n}{F}$ in \mathcal{R} si ha che $F_1, \dots, F_n \vdash F$ usando solamente le regole in \mathcal{S} .

La derivabilità ci permette, per risolvere un problema, di ridurci a risolverne un altro, infatti il **teorema della derivabilità** dice che:

se \mathcal{R} è derivabile a partire da \mathcal{S} allora per ogni dimostrazione ottenuta usando solo regole in \mathcal{R} esiste una dimostrazione con le stesse premesse e conclusione che usa solo regole in \mathcal{S} .

8.7 Regole di dimostrazione

Le regole di dimostrazione sono corrette rispetto a una certa logica che va dichiarata.

8.7.1 Introduzione di \wedge

Regola \wedge_i :

$$\frac{F_1 \quad F_2}{F_1 \wedge F_2}$$

Lettura bottom-up: se vale F_1 e vale F_2 allora vale $F_1 \wedge F_2$.

Lettura top-down: per dimostrare $F_1 \wedge F_2$ debbo dimostrare sia F_1 che F_2 (dividiamo la dimostrazione in due sotto-dimostrazioni).

La regola è invertibile.

8.7.2 Eliminazione di \wedge

Regola \wedge_e :

$$\frac{F_1 \wedge F_2 \quad \begin{array}{c} [F_1][F_2] \\ \vdots \\ F_3 \end{array}}{F_3}$$

Lettura bottom-up: se vale $F_1 \wedge F_2$ e se ipotizzando F_1 e F_2 concludo F_3 , allora vale F_3 .

Lettura top-down: per dimostrare F_3 data l'ipotesi $F_1 \wedge F_2$ è sufficiente dimostrare F_3 sotto le ipotesi F_1 e F_2 (che quindi saranno ipotesi scaricate)

Mettendo le ipotesi scaricate non ci si sta restringendo a dei sotto-casi, in quando tali ipotesi scaricate sono già implicite nell'ipotesi di partenza.

La regola normalmente **non è invertibile**, ma lo diventa se si assume $F_1 \wedge F_2$ (ad esempio se è un'ipotesi che abbiamo).

8.7.3 Regole alternative di eliminazione di \wedge

Esistono altre due regole di eliminazione di \wedge : in alcune logiche sono le uniche possibili.

Regola \wedge_{e_1} :

$$\frac{F_1 \wedge F_2}{F_1}$$

Regola \wedge_{e_2} :

$$\frac{F_1 \wedge F_2}{F_2}$$

Lettura bottom-up: se vale $F_1 \wedge F_2$ allora vale F_1 (F_2)

Lettura top down: per dimostrare F_1 (F_2) basta dimostrare $F_1 \wedge F_2$

Queste **non sono invertibili**.

Teorema: l'insieme di regole $\{\wedge_{e_1}, \wedge_{e_2}\}$ è derivabile a partire dall'insieme $\{\wedge_e\}$ e viceversa.

8.7.4 Introduzione di \vee

Regola \vee_{i_1} :

$$\frac{F_1}{F_1 \vee F_2}$$

Regola \vee_{i_2} :

$$\frac{F_2}{F_1 \vee F_2}$$

Lettura bottom-up: se F_1 (F_2) vale, allora vale anche $F_1 \vee F_2$ **Lettura top-down:** per dimostrare $F_1 \vee F_2$ è sufficiente dimostrare F_1 (F_2)

Le regole **non sono invertibili**: poiché avendo a disposizione due regole tra cui scegliere, non è detto che una valga sempre, ma la scelta andrà fatta a seconda delle ipotesi che si hanno.

In un determinato momento della dimostrazione potrebbero essere anche errate (non applicabili) entrambe: queste regole si usano solitamente alla fine della dimostrazione, quando si è sicuri di quale scegliere.

Diciamo quindi che esse sono **fortemente non invertibili**.

8.7.5 Eliminazione di \vee

Regola \vee_e :

$$\frac{F_1 \vee F_2 \quad \begin{array}{c} [F_1] \\ \vdots \\ F_3 \end{array} \quad \begin{array}{c} [F_2] \\ \vdots \\ F_3 \end{array}}{F_3}$$

La regola di eliminazione di \vee spezza la dimostrazione in due casi.

Lettura bottom-up: se vale $F_1 \vee F_2$ e F_3 vale sia quando vale F_1 che quando vale F_2 , allora necessariamente F_3 vale.

Lettura top-down: per dimostrare qualunque cosa sapendo $F_1 \vee F_2$ è sufficiente procedere per casi, dimostrando la stessa cosa assumendo prima che F_1 valga e poi che valga F_2 .

La regola è **parzialmente invertibile**: è invertibile se $F_1 \vee F_2$ è una ipotesi che vale.

L'aggiunta delle ipotesi scaricate nei due rami della dimostrazione è lecito in quanto alla fine globalmente si sarà comunque dimostrato per tutti i possibili casi.

8.7.6 Introduzione del \perp

Non esistono regole di introduzione del \perp .

È possibile dimostrare \perp solo se è presente tra le ipotesi o se le ipotesi sono inconsistenti.

8.7.7 Eliminazione del \perp

Regola \perp_e :

$$\frac{\perp}{F}$$

Per dimostrare il bottom a partire da F devo dimostrare F zero volte (per l'armonia: non esistono regole di introduzione del bottom)

Lettura bottom-up: dal falso segue qualunque cosa.

Lettura top-down: per dimostrare qualunque cosa posso ridurmi a dimostrare un assurdo.

Chiaramente una regola **non invertibile**: la uso solo se vedo all'avanti che si possa arrivare a dimostrare il bottom.

8.7.8 Introduzione del \top

Regola \top_i :

$$\frac{}{\top}$$

Regola assioma: per dimostrare \top non ho bisogno di dimostrare nulla: top non è una foglia ma non viene dimostrato in quanto rappresenta ciò che è sempre vero.

Lettura bottom-up: il \top è vero.

Lettura top-down: per dimostrare \top non debbo fare nulla.

La regola è ovviamente **invertibile**.

8.7.9 Eliminazione del \top

Regola \top_e :

$$\frac{\top \quad F}{F}$$

La regola è inutile: per dimostrare F eliminando \top , mi riduco a dimostrare F con nessuna ipotesi aggiuntiva. Utilizzare questa regola è sempre un de-tour.

8.7.10 Introduzione di \Rightarrow

Regola \Rightarrow_i :

$$\frac{\begin{array}{c} [F_1] \\ \vdots \\ F_2 \end{array}}{F_1 \Rightarrow F_2}$$

Lettura top-down: per dimostrare $F_1 \Rightarrow F_2$ basta assumere F_1 e dimostrare F_2 .

(**Lettura bottom-up:** se ipotizzando F_1 dimostro F_2 allora $F_1 \Rightarrow F_2$.)

La lettura bottom-up non si usa mai.

La regola è invertibile.

8.7.11 Eliminazione di \Rightarrow

Regola \Rightarrow_e (o modus ponens):

$$\frac{F_1 \Rightarrow F_2 \quad F_1}{F_2}$$

Lettura bottom-up: se vale F_1 e $F_1 \Rightarrow F_2$, allora necessariamente vale F_2 .

Non è né invertibile né parzialmente invertibile: anche se $F_1 \Rightarrow F_2$ è dimostrabile, non è detto che F_1 sia dimostrabile.

8.7.12 Introduzione del \neg

Poiché in logica classica $\neg F \equiv (F \Rightarrow \perp)$, possiamo derivare le regole del \neg come caso speciale di quelle del \Rightarrow .

Regola \neg_i :

$$\frac{\begin{array}{c} [F_1] \\ \vdots \\ \perp \end{array}}{\neg F_1}$$

Lettura bottom-up: se ipotizzando F_1 dimostro l'assurdo allora vale $\neg F_1$.

Lettura top-down: per dimostrare F_1 basta assumere F_1 e dimostrare l'assurdo.

La regola è invertibile.

8.7.13 Eliminazione del \neg

Regola \neg_e :

$$\frac{F_1 \quad \neg F_1}{\perp}$$

Lettura bottom-up: è assurdo avere sia F_1 che $\neg F_1$.

Lettura top-down: per dimostrare l'assurdo basta dimostrare qualcosa e il suo contrario.

Di fatto è una regola per dimostrare bottom.

La regola **non sarebbe invertibile** a priori (essendo un'eliminazione di un implica).

Ma se sto dimostrando bottom, o ho fatto un errore, oppure bottom è dimostrabile: in tal caso che le ipotesi siano inconsistenti è l'unica possibilità. Quindi dal momento in cui arrivo a dimostrare bottom, tutto è invertibile, perché non ho possibilità di mettermi in un vicolo cieco (se arrivo a un F non dimostrabile potrei usare la regola di eliminazione del bottom, per tornare a dimostrare bottom).

La regola diventa quindi **invertibile**, ma bisogna stare attenti a scegliere F_1 in modo corretto.

8.8 Correttezza e completezza delle regole

Il set di regole dipende dalla logica scelta. In logiche diverse le regole possono essere diverse / applicarsi in modo diverso, inoltre scegliere una logica significa scegliere un significato di "valere".

Per scegliere un significato di valere, esistono 2 approcci:

- utilitaristico: scegliamo un significato perché è utile per i fini per i quali mi serve tale regola
- filosofico: scegliamo una filosofia che ci permette un certo tipo di approccio

Utilizziamo un metodo filosofico.

Un altro requisito di un set di regole dovrebbe essere la **completezza**, ovvero: **ogni cosa vera è dimostrabile**.

In logica classica ci sono però due tautologie (conseguenza logica dell'insieme vuoto) che non possiamo dimostrare con il set di regole visto fin'ora:

- **Terzo escluso (Excluded Middle, EM):** $\vdash A \vee \neg A$
- **Ragionamento per assurdo (RAA):** $\vdash \neg \neg A \Rightarrow A$

Quindi la deduzione naturale vista fin ora non è completa rispetto alla semantica classica.

A questo punto si può procedere in due modi:

- Introdurre nuove regole
- Cercare un'altra logica in cui queste regole sono sensate. In questo caso una risposta è la logica intuizionistica, che tra l'altro è ottima per gli informatici.

Ciò che però non è dimostrabile intuisticamente (ma che classicamente lo sarebbe) ha bisogno di due nuove regole.

8.8.1 Regola di dimostrazione RAA

Regola RAA :

$$\frac{\begin{array}{c} [\neg F] \\ \vdots \\ \perp \end{array}}{F}$$

Lettura bottom-up: Assumiamo per assurdo $\neg F$. . . Assurdo! Quindi F .

Lettura top-down: Per dimostrare F procediamo per assurdo assumendo $\neg F$ e dimostrando \perp .

Questa è la "vera" dimostrazione per assurdo

Per dimostrare F io suppongo $\neg F$ e dimostro l'assurdo: in questo modo se ottengo che non esistono mondi in cui vale $\neg F$, vuol dire che valeva F

È una regola invertibile.

È meglio utilizzarla solo se non ho altre possibilità.

8.8.2 Dimostrazione tramite EM

In generale le dimostrazioni classiche effettuate con il solo ausilio della RAA possono essere laboriose e/o anti-intuitive.

Tuttavia il principio del terzo escluso (EM) combinato con l'eliminazione dell'or fornisce uno schema di prova molto potente (analisi per casi su una variabile):

$$\frac{\begin{array}{c} EM \\ \vdots \\ A \vee \neg A \end{array} \quad \begin{array}{c} [A] \\ \vdots \\ F \end{array} \quad \begin{array}{c} [\neg A] \\ \vdots \\ F \end{array}}{F}$$

Dove EM è la dimostrazione (classica) che $A \vee \neg A$ vale. Essa è possibile tramite il ragionamento per assurdo.

Dimostrazione di EM

$$\frac{\begin{array}{c} [\neg(A \vee \neg A)] \\ \hline \frac{\frac{\perp}{\neg A} \neg_i}{(A \vee \neg A)} \wedge_{i_R} \end{array} \quad \frac{\begin{array}{c} [A] \\ \hline (A \vee \neg A) \end{array} \wedge_{i_L}}{[\neg(A \vee \neg A)]} \neg_e$$

$$\frac{\frac{\perp}{A \vee \neg A} RAA}{\perp} \neg_e$$

9 Logica del prim'ordine

La logica proposizionale è quella logica le cui connotazioni possono denotare valori di verità. Però il ragionamento della logica proposizionale, spesso non è abbastanza ricco per esprimere certi concetti.

Una logica più ricca è la **logica del prim'ordine**.

9.1 Sintassi

La logica del prim'ordine ha due classi distinte di elementi:

- **Formule (proposizioni):** connotazioni che denotano valori di verità
- **Termini:** connotazioni che denotano elementi del dominio del discorso (i numeri naturali, gli animali ...)

I termini possono rappresentare una delle seguenti categorie:

- **Variabili:** elementi del dominio del discorso (usati con \forall e \exists). Di solito sono rappresentate con lettere minuscole (x, y, \dots)
- **Costanti:** elementi fissati una volta per tutte (Es. 2, \mathbb{N} , ...)
- **Funzioni:** si presentano nella forma $f^n(t_1, \dots, t_n)$, dove n rappresenta la arietà (numero di argomenti) e t_i sono gli argomenti. Le funzioni trasformano elementi (i parametri) in altri elementi del dominio del discorso. Le funzioni $f^0()$ (funzioni di arietà 0) potrebbero essere usate per rappresentare le costanti (ma spesso si usano semplicemente le costanti)

Le **proposizioni** possono rappresentare una delle seguenti categorie:

- Connettivi/simboli già presenti nella logica proposizionale: $\perp, \top, P \vee P \dots$
- **Quantificatore universale:** $\forall x.P$
- **Quantificatore esistenziale:** $\exists x.P$
- **Predicati:** si presentano nella forma $P^n(t_1, \dots, t_n)$ ed esprimono valori di verità. A differenza della logica proposizionale possono avere una qualunque arietà n , ma i predicati P^0 rappresentano le variabili proposizionali ($A, B \dots$)

La logica si chiama del prim'ordine perché quando usiamo \forall e \exists possiamo usare solo le variabili che rappresentano elementi del dominio del discorso ($\forall x.F$), e non funzioni, predicati ...

Nota: $f(x)$ non è un'applicazione di funzione in senso logico quando, come in teoria degli insiemi, rappresenta un abuso di notazione per indicare quell'unico y t.c. $\langle x, y \rangle \in f$, in quanto f in questo caso rappresenta un insieme (di coppie tali che ...) e non una funzione logica.

9.2 Semantica classica della logica del prim'ordine

Un mondo/interpretazione v fissa un insieme non vuoto A e assegna:

- a ogni costante c un elemento di A
- a ogni simbolo di funzione n -aria f^n , una funzione n -aria su A : $[[f^n]]^v \in A^{A \times \dots \times A}$
- a ogni simbolo di predicato n -ario P^n , un predicato n -ario su A : $[[P^n]]^v \in \{0, 1\}^{A \times \dots \times A}$

Quindi il significato dei simboli non è fissato, non è unico, ma può variare a secondo del mondo (magari in un mondo v , vale che $[[+]]^v = *$, ovvero la semantica dell'addizione nel mondo v è quella della moltiplicazione)

Semantica di \forall e \exists :

- $[[\forall x.F]]^v = 1$ sse la semantica di F nel mondo v è **sempre 1** al variare della semantica di x su tutti gli elementi di A
- $[[\exists x.F]]^v = 1$ sse la semantica di F nel mondo v è **almeno una volta 1** al variare della semantica di x su tutti gli elementi di A

Nota: questa semantica non è più un algoritmo in quanto lavoro su insiemi infiniti

9.3 Binder

In generale un binder **lega** una **variabile** ad uno **scope**. Esistono vari tipi di binder in matematica, informatica, logica...

I quantificatori \forall, \exists sono casi particolari di **binder**.

Per il \forall e \exists lo scope della variabile che lega va dalla dichiarazione di variabile (accanto al simbolo \forall/\exists), alla fine della formula associata. Tale variabile si dice quindi che è **legata** al quantificatore.

Nel caso di scope innestati, avviene lo **shadowing** se viene ri-dichiarata una variabile con lo stesso nome.

Esempio

Consideriamo la formula

$$\forall x.(P(x) \wedge \exists x.Q(x))$$

Lo scope associato al \forall è l'intera formula $P(x) \wedge \exists x.Q(x)$.

Lo scope associato all' \exists invece è la formula $Q(x)$.

Poiché l' \exists ri-lega la variabile x , la x che occorre in $Q(x)$ si riferisce (è legata) al quantificatore esistenziale.

Non è più possibile riferirsi alla x legata dal \forall nello scope dell' \exists .

Questo fenomeno è proprio lo **shadowing**.

Per questo si possono introdurre i **diagrammi di legame** (informali).

Per ottenere il relativo diagramma di legame, scritta un'espressione:

- collegare ogni occorrenza di una **variabile legata** con il binder che la lega, per mezzo di una freccia
- le variabili non legate sono dette **libere**: indicarle con una freccia che punta all'infinito su cui scrivete il **nome della variabile**

Quello che determina la semantica di una formula è il diagramma di legame, non i nomi delle variabili. Quindi due formule scritte usando nomi diversi per le variabili possono essere la stessa formula se hanno lo stesso diagramma di legame.

Definizione delle **variabili libere** (free variables) per ricorsione:

$$\begin{aligned} FV(x) &:= \{x\} \\ FV(f^n(t_1, \dots, t_n)) &:= FV(t_1) \cup \dots \cup FV(t_n) \\ FV(P^n(t_1, \dots, t_n)) &:= FV(t_1) \cup \dots \cup FV(t_n) \\ FV(\perp) = FV(\top) &:= \emptyset \\ FV(\neg P) &:= FV(P) \\ FV(P \wedge Q) = FV(P \vee Q) = FV(P \Rightarrow Q) &:= FV(P) \cup FV(Q) \\ FV(\forall x.P) = FV(\exists x.P) &:= FV(P) \setminus \{x\} \end{aligned}$$

In particolare è importante notare come i quantificatori catturano la variabile (x), il che la rende non più una variabile libera.

9.4 α -convertibilità

I nomi scelti per le variabili legate non hanno alcuna importanza se generano lo stesso diagramma di legame.

Le formule equivalenti (=stesso diagramma di legame) si dicono **α -convertibili**. E rappresentano proprio la stessa formula.

Bisogna notare però che a volte cambiare i nomi delle variabili porta a formule non α -convertibili!

Esempio

La formula

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

è equivalente a

$$\forall z.P(z) \wedge \exists y.P(z, y)$$

mentre non è equivalente (a causa dello shadowing) a

$$\forall z.P(z) \wedge \exists z.P(z, z)$$

Per essere più formali quando si lavora con una formula, bisognerebbe invece lavorare con la rispettiva classe di equivalenza per \equiv_α .

9.5 Sostituzione

Il senso del quantificatore universale $\forall x.P(x)$ è che da esso dobbiamo essere in grado di dedurre $P(t)$ per un qualunque termine t .

Serve quindi una **funzione di sostituzione** di un t al posto di una variabile x precedentemente legata. Nel fare questo bisogna però fare attenzione a non modificare il diagramma di legame della formula.

Utilizziamo la notazione $[t/x]$ a destra della formula per indicare che mettiamo t al posto di x . Questo diventa uno specie di binder che cattura tutte le variabili libere di nome x con scope tutta la formula a sinistra.

La sostituzione di un termine per una variabile nelle formule che prevedono \forall e \exists si comporta nel seguente modo (per induzione strutturale sul termine in cui avviene la sostituzione):

$$\begin{aligned} (\forall x.P)[t/x] &= \forall x.P \\ (\forall y.P)[t/x] &= \forall z.(P[z/y][t/x]) \text{ per } z \notin FV(t) \cup FV(P) \\ (\exists x.P)[t/x] &= \exists x.P \\ (\exists y.P)[t/x] &= \exists z.(P[z/y][t/x]) \text{ per } z \notin FV(t) \cup FV(P) \end{aligned}$$

In pratica quando eseguiamo una sostituzione in una formula che presenta un binder, dobbiamo anche cambiare la variabile legata al binder (se necessario), in modo che essa non appartenga alle variabili libere del termine t che abbiamo scelto per la spedizione né alle variabili libere di P .

Per evitare problemi è sempre meglio scegliere una variabile fresca.

Una variabile **fresca** è una variabile mai usata prima: si introduce per sostituire i nomi di altre variabili che sennò causerebbero problemi

10 Deduzione naturale per la logica del prim'ordine

La logica del prim'ordine estende le proposizioni della logica proposizionale in due modi:

- Il caso P^0 è generalizzato a $P^n(t_1, \dots, t_n)$: nessuna nuova regola necessaria, ovvero ogni proposizione P si comporterà come una variabile proposizionale nella logica proposizionale.
- Vengono aggiunti i **quantificatori universale ed esistenziale**: è sufficiente introdurre le relative regole di introduzione ed eliminazione.

Una cosa che capita spesso in logica del prim'ordine è cambiare il nome alle variabili, quando la formula mantiene la relazione di equivalenza \equiv_α . Questo non è un passaggio logico, è sempre possibile farlo, senza usare nessuna regola in particolare.

Quando effettuiamo un cambio di variabile sempre meglio scegliere una **variabile fresca**.

10.1 Introduzione del \forall

Regola \forall_i :

$$\frac{\begin{array}{c} \vdots \\ P[y/x] \end{array}}{\forall x.P}$$

Dove $y \notin FV(Foglie(:))$ e dove $Foglie(:)$ sono le foglie non (ancora!) cancellate nel sotto-albero :

In pratica bisogna verificare di prendere una variabile y che non appartenga alle variabili libere dell'albero. Dopodiché si passa a dimostrare P sostituendo y al posto di x .

Importante: y deve essere una variabile, non una costante.

Lettura bottom-up: se P vale per una y qualunque (scelta arbitrariamente), allora per ogni x vale P .

Lettura top-down: per dimostrare $x.P$ è sufficiente scegliere una y sulla quale non sappiamo nulla e poi dimostrare $P[y/x]$

La regola è **invertibile**. Applicarla subito ogni volta.

10.2 Eliminazione del \forall

Regola \forall_e :

$$\frac{\forall x.P}{P[y/x]}$$

Lettura bottom-up: se P vale per tutti gli x , allora vale in particolare per y .

Lettura top-down: per dimostrare $P[y/x]$ è sufficiente dimostrare il teorema generalizzato $\forall x.P$

La regola è chiaramente **non invertibile**.

10.3 Introduzione del \exists

Regola \exists_i :

$$\frac{P[y/x]}{\exists x.P}$$

Lettura bottom-up: se P vale per y allora esiste un x per cui P vale.

Lettura top-down: per dimostrare $\exists x.P$ bisogna scegliere un y per il quale $P[y/x]$ valga e dimostrarlo.

La regola è ovviamente non invertibile. **Fortemente non invertibile**.

10.4 Eliminazione del \exists

Regola \exists_e :

$$\frac{\exists x.P \quad \begin{array}{c} [P[y/x]] \\ \vdots \\ C \end{array}}{C}$$

Dove $y \notin FV(Foglie(:)) \cup FV(C)$ e dove $Foglie(:)$ sono le foglie non (ancora!) cancellate nel sotto-albero :

Lettura bottom-up: se $\exists x.P$ e se dimostro C sotto l'ipotesi che P valga per un generico y , allora C vale.

Lettura top-down: per dimostrare un qualche C sotto l'ipotesi $\exists x.P$ è sufficiente dimostrare C assumendo P per una qualche variabile generica y .

La regola è **parzialmente invertibile**: la usiamo subito se abbiamo l'ipotesi $\exists x.P$

11 Ricorsione e induzione

11.1 Introduzione

Iniziamo osservando un po' di differenze tra la matematica e l'informatica.

In matematica:

- Ogni entità è esprimibile usando insiemi
- Alcune entità possono avere solo descrizioni infinite (es. i numeri reali)
- Una funzione è un'insieme di coppie (dominio \rightarrow codominio) e se tale insieme è infinito non mi dà nessuna procedura di calcolo

In informatica:

- Un dato ha sempre una descrizione finita in quanto deve stare in memoria (che è finita)
- L'insieme dei dati è sempre enumerabile: la memoria è una sequenza di bit, tutti i dati devono stare in quella sequenza
- Un dato può però avere una dimensione **unbounded**. La dimensione è unbounded quando la dimensione del singolo dato è finita, ma per ogni dimensione arbitraria è possibile creare un dato di quella dimensione
- Un dato unbounded ha sempre **struttura ricorsiva**: all'interno di un dato più grande compare un dato (dello stesso tipo) più piccolo. Infatti se il dato fosse caotico non si potrebbe creare un unico codice per analizzarlo (che funzioni su dati di dimensione arbitraria), ma servirebbe un codice unbounded, il che non sarebbe possibile.

Esempio

$$\mathbb{N} ::= O \mid S \mathbb{N}$$

Ovvero: “un numero naturale o è lo zero (costruttore O) o è il successore di un numero naturale n (costruttore $S \ n$)”

- Un programma o funzione (ma nel senso informatico) descrive una procedura di calcolo che partendo dall'input dà un output

In informatica dato un programma f , si può sempre definire la funzione matematica associata definita come:

$$\{\langle i, o \rangle \mid f(i) = o\}$$

ovvero tutte le possibili coppie input-output tali che $f(input) = output$.

Affinché un programma finito (es. 10 linee) possa processare un input di dimensione unbounded (es. di taglia 10, 100, 1000, 10000, ...) il codice del programma deve essere eseguito ripetutamente.

Se il linguaggio di programmazione è **imperativo**, si può mutare il valore delle variabili, quindi possiamo usare strutture ciclo (**while**, **for**, ...).

Nei **linguaggi di programmazione funzionali** invece non esiste l'assegnamento (cercano di avvicinarsi al linguaggio matematico), quindi non esistono i cicli: per eseguire più volte lo stesso codice, si usano chiamate **ricorsive**.

Tutte le volte che il codice può ripetersi si rischia però la **divergenza**: ovvero quando, dato un certo input, il programma non termina mai.

Presto vedremo la **ricorsione strutturale**: ricorsione vincolata che non diverge mai.

11.2 Tipi di dato

In un linguaggio di programmazione funzionale non esistono dati come:

- Puntatori
- Classi
- Array
- ...

In un linguaggio di tipo funzionale invece gli unici tipi di dati disponibili sono un **Tipo di Dato Algebrico (ADT)**, che sono definiti da:

- un **nome** del tipo

- una lista di **possibili forme** (o **costruttori**) per quel tipo
- ogni dato può contenere altri dati specificandone il loro tipo
- la **ricorsione** è ammessa, ovvero i sotto-dati possono essere della stessa forma

Ogni **valore** è rappresentato da un **albero** in cui nodi sono una possibile formula e i cui sotto-alberi sono altri valori.

Se un dato non è definito con la ricorsione, viene anche detto dato **enumerato** (es. i booleani)

Esempi di tipi di dati algebrici

Booleani:

$$\mathbb{B} ::= \text{tt} \mid \text{ff}$$

Es: tt rappresenta il vero, ff il falso.

Numeri naturali:

$$\mathbb{N} ::= O \mid S\ N$$

Es: $S(S(S(SO)))$ è il numero 4.

Liste di numeri naturali:

$$\mathbb{L} ::= [] \mid \mathbb{N} :: \mathbb{L}$$

Es. $(SO) :: (O :: ((S(SO)) :: []))$ è la lista [1, 0, 2]

In generale una lista può essere o la lista vuota, o una testa (un numero naturale), seguito da una lista.

L'operatore $::$ viene detto "**cons**" e rappresenta il costruttore.

Anche le formule della logica proposizionale e gli alberi sono tipi di dato algebrici

Generalizzazioni:

1. **Polimorfismo:** un dato (e tutte le sue funzioni associate) è polimorfo quando si può parametrizzare rispetto a un tipo generale (come con i template in c++), ma una volta istanziato il dato tutti i valori devono essere dello stesso tipo.
2. **Tipi di dato algebrici mutui:** due dati che devo definire contemporaneamente perché dipendono uno dall'altro (es. numeri pari e dispari). Per lavorarci devo avere coppie di funzioni (funzioni mutue). Anche gli alberi n-ari sono mutui con le liste (di alberi).

11.3 Programmi

I programmi sono costituiti da liste di funzioni e hanno solo tipi di dato algebrici.

La definizione di una funzione è fatta da tante righe del seguente tipo:

$$f(w_1, x_1, \dots, x_m) = \dots \text{corpo}_1 \dots$$

\vdots

$$f(w_n, x_1, \dots, x_m) = \dots \text{corpo}_n \dots$$

- f è il nome della funzione ($m + 1$)-aria ($m = 0$ ammesso)
- w_i è un **pattern**: è una forma possibile dell'input (un costruttore). Servono tante righe quanti costruttori ci sono
- x_1, \dots, x_n sono parametri formali (come in un qualsiasi linguaggio di programmazione)
- corpo_i è l'output associato: può contenere
 - forme di tipi di dato
 - parametri formali
 - if-then-else
 - chiamate a funzione

Quali funzioni si possono usare (all'esame)?

- Tutte quelle ovviamente implementabili e non interessanti ai fini dell'esercizio (p.e. operatori algebrici $+$, $*$, $=$, \leq , \dots , operatori booleani $\&\&$, $\|$, \dots , etc.)
- Tutte quelle definite precedentemente nell'esercizio
- Una funzione f può invocare la stessa f (chiamata ricorsiva); in tal caso la f si dice **funzione ricorsiva**. Si possono usare solo funzioni **strutturalmente ricorsive**

11.4 Pattern matching

Dato un pattern ω e un dato algebrico E , ω fa match con E se è possibile risolvere l'equazione $\omega = E$ sostituendo ai parametri formali contenuti in ω delle sotto-espressioni di E .

Esempio

$\langle x, y \rangle$ fa match con $\langle O :: [], tt \rangle$ attraverso la sostituzione $[(O :: [])/x ; tt/y]$
 $\langle x, y \rangle$ non fa match con $O :: SO :: []$

L'invocazione di una funzione avviene per **pattern matching**. Supponendo che ci sia una e una sola dichiarazione $f(w_i, x_1, \dots, x_m) = \dots \text{corpo}_i \dots$ tale che w_i faccia match con E_0 , allora la chiamata $f(E_0, E_1, \dots, E_m)$ viene riscritta in corpo_i dopo aver sostituito a ogni parametro formale del pattern la soluzione dell'equazione $\omega = E$ e a ogni **parametro formale** x_j il parametro attuale (espressione) E_j .

Nota: nel caso più dichiarazioni di funzione facciano matching con una chiamata di funzione potrebbe esserci una soluzione **non deterministica**.

11.5 Ricorsione strutturale

I tipi di dato algebrici hanno struttura ricorsiva, ovvero un elemento o è **atomico**, o è composto a partire da parti più piccole, alcune con la stessa struttura.

Su strutture di questo tipo è possibile risolvere un problema con una funzione per **ricorsione strutturale**:

- Abbiamo una funzione (dichiarazione) per ogni costruttore del tipo di dato su cui vogliamo ricorrere (la f deve considerare come pattern tutte le forme del dato una e una sola volta).
- Se il costruttore è atomico, la soluzione è semplice e si risolve direttamente (restituiamo direttamente l'output)
- Nei casi composti si risolve prima il problema **sulle componenti** (chiamata ricorsiva sulla parte più piccola del dato) e poi si sintetizza la risposta per il caso composto, avendo a disposizione la soluzione per le componenti più piccole.
- Quando la funzione viene chiamata ricorsivamente, bisogna passare come primo parametro attuale solamente uno dei parametri formali contenuti nel pattern

Teorema

Tutte le funzioni definite per ricorsione strutturale convergono su ogni input.

Esempio di ricorsione strutturale:

```
1 -- Problema 5: dato un numero x e una lista di liste di numeri LL inserire x in testa a ognuna delle liste di LL
2
3 def inserth : Nat -> List (List Nat) -> List (List Nat)
4 | x, [] => []
5 | x, L::LL => (x :: L) :: (inserth x LL)
6
7 #eval inserth 1 [[2,3],[4,5]]
8
9 -- Problema 4: date due liste concatenarle
10 -- Esempio:
11 -- conc [1,2] [3,4] = [1,2,3,4]
12
```

```

13 def conc : List X -> List X -> List X
14 | [], _L => _L
15 | h::tl, _L => h :: (conc tl _L)
16
17 #eval conc [1,2] [3,4]
18
19 -- Problema 3: dato un numero x e una lista di numeri L, restituire la lista di lista ottenuta inserendo x
    in tutte le possibili posizioni di L
20 -- Esempio:
21 -- insert 1 [2,3] = [[1,2,3],[2,1,3],[2,3,1]]
22
23 def insert : Nat -> List Nat -> List (List Nat)
24 | x, [] => [[x]]
25 | x, h::L => (x::h::L) :: inserth h (insert x L)
26
27 #eval insert 1 [2,3]
28
29 -- Problema 2: dato un numero x e data una lista di liste LL restituire la lista ottenuta inserendo x in
    tutte le possibili posizioni in ognuna delle liste di LL
30 -- Esempio:
31 -- insertll 1 [[2,3]], [3,2] = [[1,2,3],[2,1,3],[2,3,1],[1,3,2] ... ]
32
33 def insertll : Nat -> List (List Nat) -> List (List Nat)
34 | x, [] => []
35 | x, L::LL => conc (insert x L) (insertll x LL)
36
37 #eval insertll 1 [[2,3],[3,4]]
38
39 -- Problema 1: data una lista di naturali L restituire come lista l'insieme di tutte le permutazioni di L
40 -- Esempio:
41 -- permut [1,2,3] = [[1,2,3],[1,3,2],[2,1,3] ... ]
42
43 def permut : List Nat -> List (List Nat)
44 | [] => [[]] -- [] :: []
45 | h::tl => insertll h (permut tl)
46
47 #eval permut [1,2,3]

```

11.6 Induzione strutturale

Come si dimostra che una funzione definita per ricorsione strutturale gode di una certa proprietà? Usando l'**induzione strutturale**!

Sia P una proprietà che vogliamo dimostrare valere su tutti i valori di un tipo di dato algebrico. In logica del prim'ordine dovremmo usare la regola del \forall (ovvero usare un input generico). L'induzione strutturale invece ci permette di ragionare per ricorsione strutturale. La dimostrazione viene quindi data in questo modo:

- Una sotto-dimostrazione per ogni forma del tipo di dato (pattern).
- In ogni sotto-dimostrazione possiamo assumere che P già valga su tutti i parametri formali contenuti nel pattern (**ipotesi induttive**). Non è una vera assunzione, infatti è la chiamata ricorsiva ci dà tale ipotesi, non ci si sta restringendo a dei casi particolari (come avviene invece nel caso di un'implicazione).

Una dimostrazione per induzione strutturale prevede un certo numero di **fasi**:

Sempre all'inizio della dimostrazione:

1. **Fase di proclamazione:** esplicitiamo che procediamo per induzione strutturale identificando un certo input per dimostrare la proprietà P (su tale dato). In questa fase bisogna osservare le funzioni e capire quale input utilizzare: in generale si sceglie quello che viene utilizzato maggiormente (velocizza la dimostrazione). "procedo per induzione strutturale..." diventa un binder per quello che andiamo a dimostrare (la variabile che rappresenta il dato su cui ricorriamo)

Poi per ogni caso (pattern):

2. **Fase dell'ipotesi induttiva:** assumiamo che la proprietà P valga per per la componente ricorsiva (più piccola) del dato su cui sto lavorando per induzione. Nel caso ci troviamo nel caso di un pattern atomico, questa fase non viene effettuata.

3. **Fase di enunciazione:** enunciamo cosa dobbiamo dimostrare, dopo aver fatto la sostituzione del dato generale, con il pattern del caso in cui ci troviamo.
4. **Fase di semplificazione:** avendo in input casi particolari e non più generali, possiamo espandere le definizioni, osservando come il programma calcola in tali casi, andando così a espandere e semplificare ciò che dobbiamo dimostrare.
5. **Fase di prova:** concludiamo la prova con un'usuale dimostrazione in logica del prim'ordine.

Esempio di esercizio:

Considerare alberi binari definiti dalla grammatica:

$$T ::= N \mid \langle T, N, T \rangle$$

Considerare le seguenti funzioni:

$$\begin{aligned} \text{sum}(n) &= n \\ \text{sum}(\langle T, n, T \rangle) &= \text{sum}(L) + n + \text{sum}(R) \\ \text{prune}(n) &= n \\ \text{prune}(\langle T, n, T \rangle) &= \langle \text{if } \text{sum}(\text{prune}(L)) = 0 \text{ then } 0 \text{ else } \text{prune}(L), \\ &\quad n, \\ &\quad \text{if } \text{sum}(\text{prune}(R)) = 0 \text{ then } 0 \text{ else } \text{prune}(R) \rangle \end{aligned}$$

Dimostrare, per induzione strutturale, che $\forall T. \text{sum}(\text{prune}(T)) = \text{sum}(T)$

Lemma:

$$\forall T. \text{sum}(\text{if } \text{sum}(T) = 0 \text{ then } 0 \text{ else } \text{prune}(T)) = \text{sum}(\text{prune}(T))$$

Dimostrazione:

Sia T un albero.

Procediamo per casi su $\text{sum}(T) = 0 \vee \text{sum}(T) \neq 0$:

Caso $\text{sum}(T) = 0$:

Devo dimostrare $\text{sum}(0) = \text{sum}(\text{prune}(0))$, che è equivalente a $0 = 0$.

Ovvio per proprietà riflessiva dell'uguaglianza.

Caso $\text{sum}(T) \neq 0$:

Devo dimostrare $\text{sum}(\text{prune}(T)) = \text{sum}(\text{prune}(T))$.

Ovvio per proprietà riflessiva dell'uguaglianza.

Teorema:

$$\forall T. \text{sum}(\text{prune}(T)) = \text{sum}(T)$$

Dimostrazione:

Procedo per induzione strutturale su T per dimostrare $\text{sum}(\text{prune}(T)) = \text{sum}(T)$.

Caso $T = n$:

Devo dimostrare $\text{sum}(\text{prune}(n)) = \text{sum}(n)$,

che è equivalente a $\text{sum}(n) = n$, che è equivalente a $n = n$.

Ovvio per proprietà riflessiva dell'uguaglianza.

Caso $T = \langle L, n, R \rangle$:

Per l'ipotesi induttiva so che $\text{sum}(\text{prune}(L)) = \text{sum}(L)$ (I_L) e che $\text{sum}(\text{prune}(R)) = \text{sum}(R)$ (I_R).

Devo dimostrare $\text{sum}(\text{prune}(\langle L, n, R \rangle)) = \text{sum}(\langle L, n, R \rangle)$,

che è equivalente a

$$\text{sum}(\langle (\text{if } \text{sum}(\text{prune}(L)) = 0 \text{ then } 0 \text{ else } \text{prune}(L)), n, (\text{if } \text{sum}(\text{prune}(R)) = 0 \text{ then } 0 \text{ else } \text{prune}(R)) \rangle) = \text{sum}(L) + n + \text{sum}(R),$$

che è equivalente a

$sum(\text{if } sum(prune(L)) = 0 \text{ then } 0 \text{ else } prune(L)) + n + (\text{if } sum(prune(R)) = 0 \text{ then } 0 \text{ else } prune(L)) = sum(L) + n + sum(R)$.

Per il lemma appena dimostrato posso ridurmi a dimostrare $sum(prune(L)) + n + sum(prune(R)) = sum(L) + n + sum(R)$.

Per I_L e I_R posso ridurmi a dimostrare $sum(L) + n + sum(R) = sum(L) + n + sum(R)$.

Ovvio per proprietà riflessiva dell'uguaglianza.