Logica per informatica

NOTE PER IL LIBRO

Questo libro è una raccolta degli appunti di **Angelo Huang**, Telegram: @flecarts per il corso di Logica dell'informatica (73149) al corso di Laurea in Informatica a Bologna per l'anno scolastico 2021/2022 con il professor coen.

Si può trovare la versione originale in NOTION a questo link

0 Introduzione

Lo scopo della logica è

- **Correttezza** del ragionamento, anche verificata attraverso algoritmi predittivi.
  - Si svilupperanno linguaggi logici
  - I metodi per la veridicità di una sentenza.

- **Possibilità** e metodi del ragionamento logico
- **Completezza e non-deducibilità** di alcuni ragionamenti
    - Necessità di completezza delle ipotesi: più ipotesi = ragionamento valido?
    - Completezza delle tesi, impossibile.

Una necessità della logica è Meta-logica:

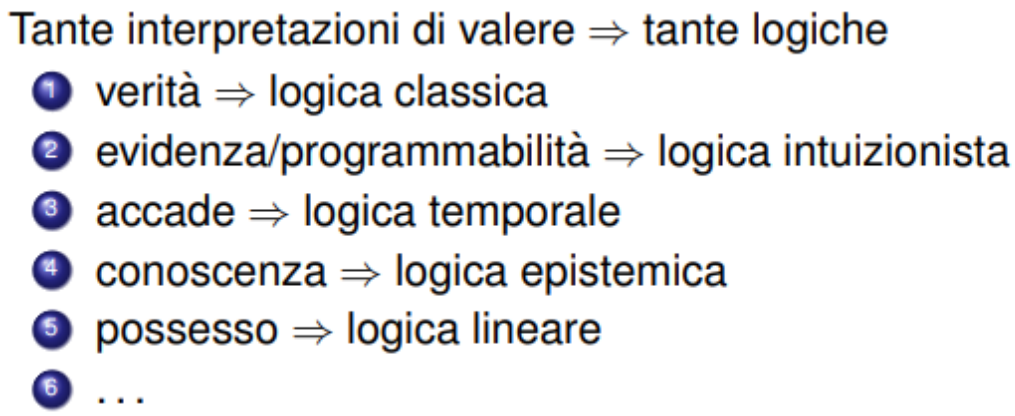
La logica si deve cercare di basare su certe basi, spesso queste non sono certe, però danno un certo grado di sicurezza → Se la base è solida allora tutto il ragionamento di una parte è giusta

0.1 Storia

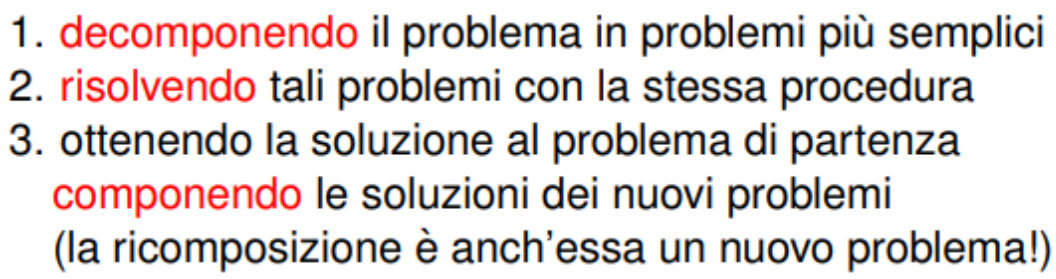
Questo campo di studi è nato dopo una necessità del secolo precedente quando si tentava di dare delle basi solide alla matematica → La matematica (e informatica)si fonda sulla logica.

Guardare la storia di Russel, Godel.

0.2 Tipologie di logica



0.3 Processo di ragionamento



Slide 18 per esempio di problema risolto con questo processo.

0.4 Differenze con la matematica

La matematica si interessa principalmente sull'**esistenza di soluzioni**, e dimostrazioni, ma non rigorose quanto le dimostraizoni logiche. L'informatica applicata classica è meno rigorosa, si basa principalmente sui test, anche se un logico può dimostrare la correttezza di una soluzione informatica.

Inoltre l'informatica indaga la possibilità di implementazione di alcune soluzioni matematiche, cioè il modo per calcolare possibili soluzioni, anche con la limitatezza delle risorse.

0.5 Logica in programmazione

0.5.1 Usi e costi

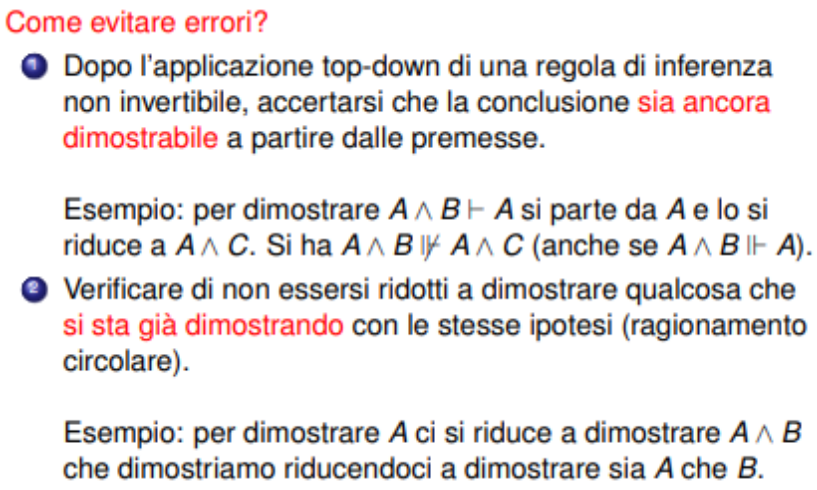
C'è un **altissimo costo per la dimostrazione formale** di un programma, secondo i dati Intel c'è bisogno di circa *10x mesi uomo* per creare una dimostrazione assistita di questo genere.

Per questo genere viene utilizzato **solamente in software critico** cioè il codice che controlla processi che se buggati possono creare ingenti danni economici, come centrali nucleari, smartcard, microprocessori Intel, controlli aereo e simili

0.5.2 Processo di dimostrazione

1. Definire la **specifica del software** in modo che possa fare sempre ciò che deve fare
2. Creare la **semantica** del programma, come il programma è eseguito.
3. **Formula logica** che è la descrizione formale del funzionamento del programma.
4. Creare una **implicazione** fra ciò che il software deve fare secondo la semantica e ciò che veramente fa.

Non perdere punti



1 Paradossi Metalinguistici

1.1 Antinomie e Paradossi

1.1.1 Antinomia

Definizione di antinomia è un ragionamento corretto da cui deriva una conclusione errata, probabilmente è **l'insieme o campo in cui stiamo operando ad essere errato** e bisogna cercare di ridefinirlo in modo più corretto, in quanto **le premesse erano accettabili**

1.1.2 Paradosso

Paradosso quando il ragionamento corretto va contro l'intuizione, come il paradosso dei gemelli in fisica e simili. **premesse erano accettabili**

Falsi paradossi: in cui c'è un errore del ragionamento da cui viene dedotto un ragionamento errato.

1.2 Linguaggio naturale

Da ora NL = Natural Language

1.2.1 Caratteristiche del NL

Il linguaggio naturale è il linguaggio comunemente utilizzato come italiano, inglese arabo e cinese etc. utilizzato nella maggior parte della vita quotidiana.

Questo linguaggio non è utile per i ragionamenti rigorosi come descrizione del calcolo o dimostrazioni in quanto questo linguaggio è:

- Fortemente dipendente dal contesto
- Ambigua grammatica: e.g. Il poliziotto ha ucciso il ladro con la pistola (pistola mezzo oppure compagnia?)

1.2.2 Paradossi in NL

Paradossi visti in classe:

- Io mento
- eterologico è eterologico.

Le cause individuate per i paradossi sono

1. Utilizzo meta-linguistico, ce si riferisce sul linguaggio stesso (Io mento).
    a. Linguaggio naturale potrebbe essere così ampio che può parlare di sé stesso, per esempio:
    contare numero di sillabe o parole, o mischiare il senso del linguaggio o simile, questo genera paradossi.
    b. Sarebbe difficile esprimere idee senza la negazione in poche parole.

    Per scoprire l'utilizzo meta linguistico si utilizza un teorema di invarianza delle denotazioni
2. Auto applicazione di meta-linguistico a se stesso (eterologico è eterologico).
    a. Cerco di usare qualcosa su sé stesso, anche se la definizione dell'aggettivo non dovrebbe essere utilizzate in questo modo, possiamo dire che perde di senso
3. L'utilizzo della negazione (x minore non definibile in meno di 1000 parole).
    a. L'utilizzo della negazione su sé stesso e anche la negazione di sé stesso crea antinomia (paradosso NL)

1.2.3 Ricerca di un linguaggio formale

La negazione è necessaria per fare i ragionamenti, non si può togliere.

Non si riesce a evitare di applicare una definizione su sé stessa, dopo che hai oggetto e soggetto puoi scegliere dove applicarlo, cioè è brutto evitare solamente l'applicazione a sé stesso, una volta creata la preposizione può essere usata senza questi piccoli vincoli.

I'uso metalinguistico invece si può evitare, e quindi bisogna abbandonare il linguaggio naturale e approdare in un linguaggio artificiale, il linguaggio rigoroso della matematica.

1.3 Linguaggio Matematico

1.3.1 Storia dell'insiemistica

Quando ancora la matematica non era ancora scienza diversa dalla informatica, i matematici si mettevano proprio a calcolare modi di calcolo,

Vennero studiate le basi e introdotte la teoria degli insiemi, da Cantor, una base per tutta la matematica attuale, ma questo viene messo in crisi dal paradosso di Russel, creando due filosofie matematiche, gli insiemisti e altri contrari.

Paradosso di Russel: $X = \{Y \mid Y \notin Y\}$ e si ha ancora un paradosso auto-refere

Dopo tutta questa diatriba, venne creato circa nel 1930 il significato di calcolare, credo e venne creato il campo dell'informatica.

1.3.2 Paradosso di Russel

$$R = \{x \mid x \notin x\}$$

Analisi del paradosso

Possiamo vedere che la teoria degli insiemi è possibile creare paradossi:

1. Presente l'uso della negazione
2. La negazione, la caratteristica usata è fatta in modo **autoreferenziale**
3. Gli insiemi possono contenere sé stessi, e quindi è possibile l'use meta-linguistico.

Quindi non può esistere un insieme che li contenga tutti come voleva sostenere Cantor.

$$R \in R \iff R \notin R$$

Soluzione trovata

Su questa soluzione basa l'intera matematica e non si sa se ci sono altri paradossi dentro questo. Potrebbe essere errata anche questa.

1. Limitazioni sulla creazione di insiemi che abbiano proprietà comunque → **Assioma di comprensione** deve essere gettata.
2. **Assioma di separazione** ovvero gli elementi devono essere presi da un insieme esistente e una proprietà di questo insieme (quindi posso solamente restringere un insieme).
3. È definita come separata la collezione di tutti gli insiemi, per evitare il paradosso di autoreferenzialità, **prevenire l'uso meta-linguistico.**

1.4 Paradossi Informatici

1.4.1 Esistenza di paradossi

- La composizione di funzioni, cioè l'utilizzo in modo meta-linguistico delle informazioni informatiche è necessaria
  - Sia in linguaggi funzionali, imperativi, basta saltare, quindi si può modificare un programma ~~ Insieme che contiene insieme, molto simile questa cosa.
- La negazione delle affermazioni è necessaria
- Una funzione applicata su sé stessa è presente, possibilissima l'autoreferenzialità

Per questo motivo non si può evitare il paradosso nell'ambito dell'informatica.

1.4.2 Paradosso sulle funzioni non totali

Riguardare se sei in grado di definire il significato di

> Espressività di un linguaggio
>
> Convergenza e divergenza di una funzione
>
> totalità di una funzione

Totalità significa che una funzione riesca a restituire un output in tempo finito.

$f(g) = not\,(g(g))$ è una funzione che può creare un paradosso, se analizzata si possono trovare tutti i tre ingredienti per la creazione di paradosso:

C'è la negazione, c'è l'uso meta-linguistico (composizione di funzione) e se al posto di g ci metto f c'è anche l'auto-referenzialità, creando un paradosso.

Le funzioni matematiche danno sempre risultato, invece le funzioni informatiche possono divergere.

1.4.3 Paradosso sulla divergenza

Nelle funzioni informatiche c'è una fase di calcolo ed elaborazione delle informazioni.

Nelle funzioni matematiche sono solamente una relazione fra insiemi (ossia calcolano un unico input).

Una funzione tale che $f(g,x) = true\, iff\, g(x)\downarrow$

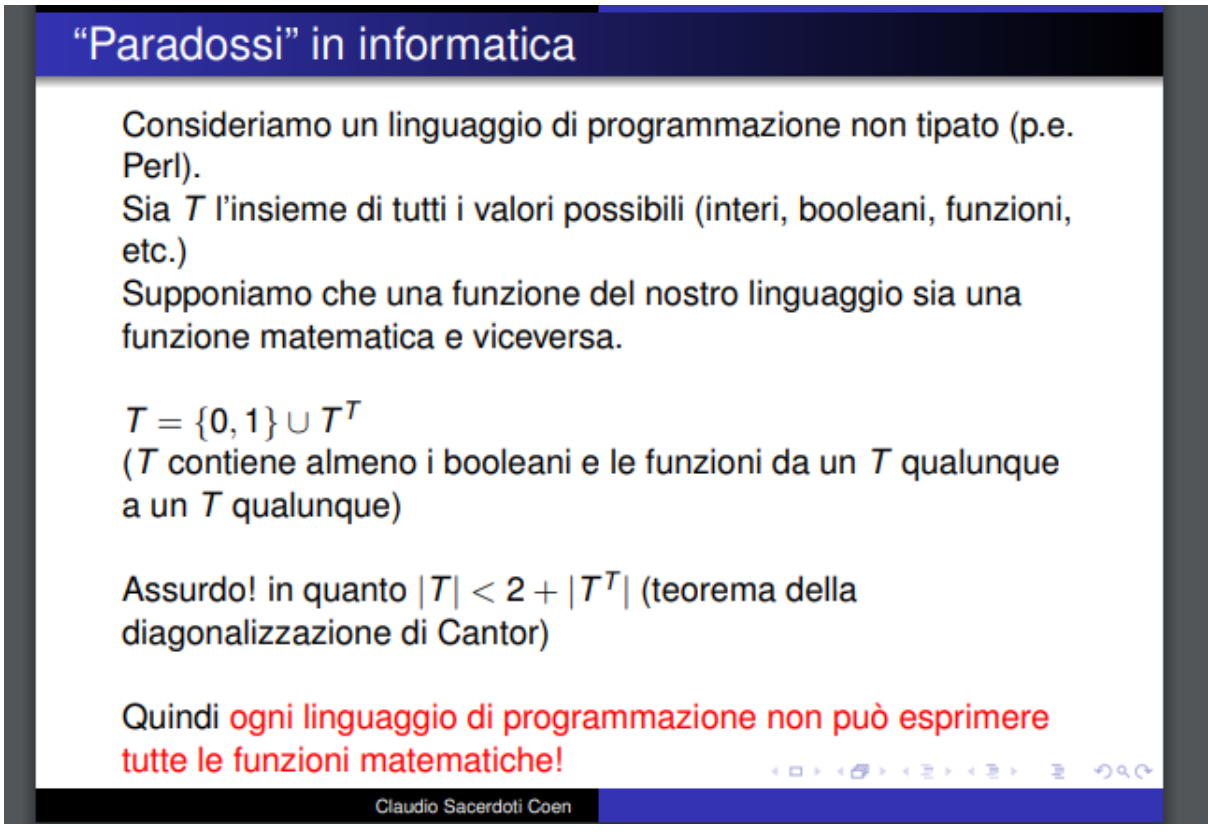
Definisco una funzione: $h(g) = \uparrow\, if\, f(g,g)\, else \downarrow$

Questo porta alla conclusione che **non esiste un programma che decida se un altro diverga**.$f(g) = \uparrow\, if\, f(g,g)\, else \downarrow$

1.4.4 Paradosso sull'espressività di funzioni matematiche

Spiegata per filo e per segno per la diagonalizzazione di cantor qui

Si risolve una assurdità attraverso il *teorema della diagonalizzazione di Cantor.*



2 Teoria degli insiemi

2.1 Elementi di base

2.1.1 Definizione e caratteristiche

1. Tutto è un insieme (su questo *si basa la maggior parte della matematica*)
2. Efficace nella descrizione degli oggetti (infiniti è ez), ma **non è efficiente** nel calcolo in quanto non dà nessun indizio sul'implementazione in memoria o sul modo per calcolarlo, c'è solo una associazione

Si può concludere che per l'informatico non serve a molto questa teoria, ma è la base per la matematica.

2.1.2 Teoria Naif

È la teoria disposta a paradossi (Russel) che afferma che gli insiemi si possono formare liberamente → **Assioma di comprensione**.

Abbiamo già analizzato in Logica meta-linguistica che questo paradosso è distruttivo,

in particolare guardare qui

Operazioni di base

- l'appartenenza
- Creazione di sottoinsiemi

2.1.3 Diagrammi di Venn

Il diagramma di Venn permette la rappresentazione di insiemi finiti

L'unica cosa di ricordare, è che **non permette nesting di insiemi**.

2.2 Zermelo-Frank Set Theory

Questa è una possibile teoria assiomatica degli insiemi.

Enti primitivi

Sono enti di cui non esiste la definizione (perché serve un punto di partenza).

In particolare sono enti primitivi

1. Appartenenza
2. Uguaglianza
    a. Nella slide c'è scritto che non vengono definiti, ma poi l'assioma di estensionalità lo definisce, invece estensionalità definisce solo una relazione
3. Insieme

Andiamo ora a definire assiomi, cose ovvie di questa teoria che sia utile per dimostrare altre proprietà

2.2.1 Assioma di Estensionalità e sottoinsiemi

$$\forall A, \forall B, \quad A = B \iff (\forall C, \quad C \in A \iff C \in B)$$

Questo assioma definisce l'uguaglianza fra gli insiemi.

Per ogni X e Y, i due insiemi sono uguali sse hanno gli stessi elementi (un Z che sta sia in X sia in Y, possiamo dire che Z è un elemento temporaneo utile per la stesura)

Sottoinsieme

Definizione = #define in c++ ossia una sostituzione, una abbreviazione.

Uguale a sopra, ma invece di SSE, utilizziamo un SE.

Se Z appartiene a X allora appartiene a Y.

Cosa è

Relazione fra l'ente primitivo uguaglianza con l'appartenenza, ma non è definito come appartenenza, ente primitivo è dato come gli pare.

2.2.2 Assioma di separazione

L'assioma di separazione è lo strumento utile che hai visto durante la risoluzione del paradosso di Russel.

Quindi definiamo un insieme Y a seconda di una caratteristica di X. Usiamo sempre un insieme temporaneo Z per definirlo.

Quindi scriviamo $Y = \{Z \in X | P(X)\}$ ossia esiste un Y definito per elementi Z tale che questo Z appartenga a X e soddisfa una proprietà P(Z). *Questo è un abuso di notazione, che però dà il senso, dovresti scrivere sempre in altro modo.*

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

Esempio, L'insieme degli studenti biondi è definito per gli studenti biondi che siano studenti (X) e che abbiano la proprietà di essere biondi P(Z).

2.2.3 Assioma dell'insieme vuoto e definizione

Questo assioma definisce l'insieme vuoto e definisce alcune caratteristiche

L'insieme vuoto non ha elementi

$\exists X, \forall Z, Z \notin X$ Ma è ridondante perché è sufficiente definirlo con l'assioma di separazione in questo modo:

Sto ottenendo l'insieme vuoto svuotandolo da un insieme che esiste già, in questo modo:

$$\emptyset \coloneqq \{X \in Y | false\}$$

2.2.4 Definizione di intersezione infinita

L'intersezione si può definire come l'insieme tale che sia contenuto sia in X sia in Y, quindi utilizzando assioma di separazione di può fare.

Teorema di appartenenza a un insieme intersezione

Si può dimostrare subito partendo dall'assioma di separazione, intendi A come insieme da cui prendi e P(Z) come appartenza a B.

Estensione a intersezioni infinite

Basta prendere l'intersezione binaria e utilizzare l'insieme di ritorno per altri, in modo ricorsivo se potrebbe aiutare.

Questo però non funziona per **l'intersezione infinita** ecco che c'è il bisogno di definire l'intersezione meglio.

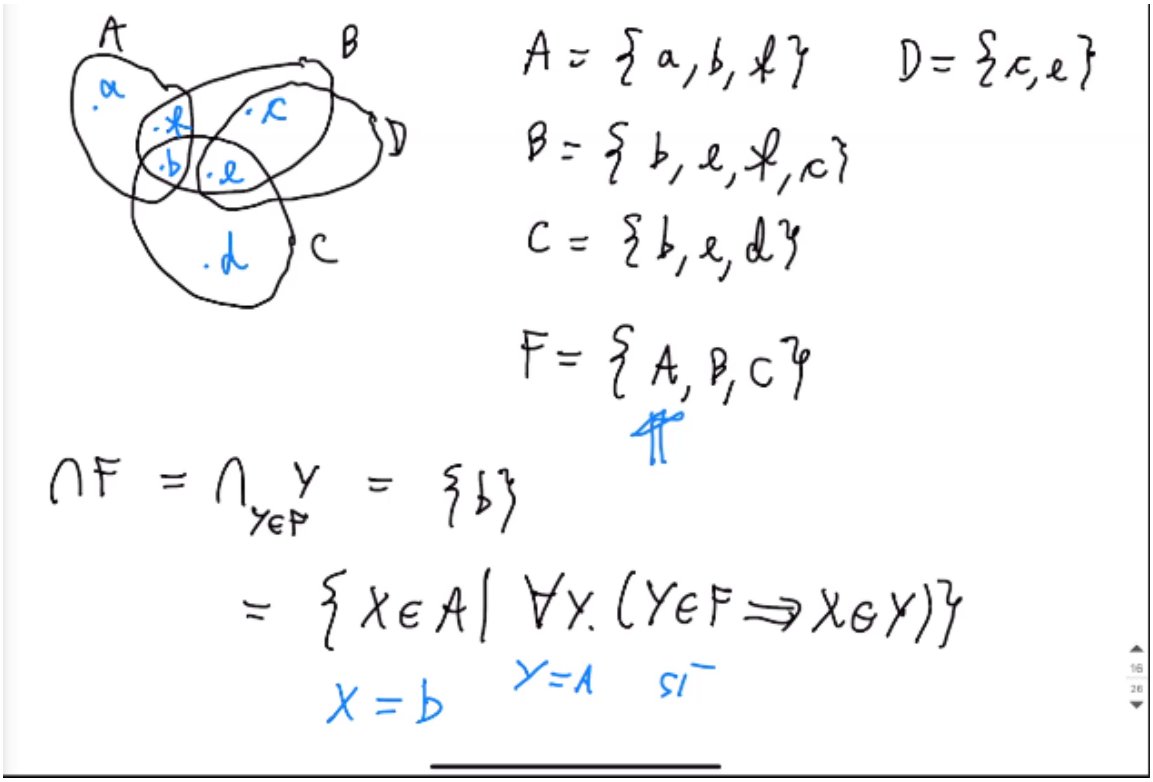
Definizione di intersezione

F è l'insieme tdi tutti gli insiemi da intersecare, se F vuoto lo definisco come vuoto, altrimenti utilizzo un insieme A e interseco sempre per ogni insieme Y!.

$$A \in F, \{X \in A | \forall Y : Y \in F \implies X \in Y\}$$

$\bigcap F =$ intersezione dell'insieme Insieme di tutti gli elementi da intersecare

▼ Esempio



2.2.5 Assioma di unione

L'assioma dell'unione definisce l'unione fra insiemi infiniti e la notazione è molto simile per l'insieme intersezione

$\bigcup F$ definito in modo simile

$$\exists A \forall X, \{X \in A \iff \exists Y : Y \in F \land X \in Y\}$$

E da questo si può dimostrare il teorema dell'unione binaria che è la definizione di unione classica che abbiamo.

2.2.6 Assioma del singoletto

Se c'è solo un elemento in un insieme allora questo si dice singoletto

$$\forall X, \exists Y, \forall Z \{Z \in Y \iff Z = X\}$$

Si può utilizzare insieme all'unione per relazionarsi all'ente primitivo dell'appartenenza.

NOTA:

l'assioma del singoletto a volte è ridondante perché si potrebbe definire in altro modo, in particolare attraverso l'assioma di rimpiazzamento

INSIEME A UNIONE

Possiamo utilizzare un abuso di notazione di questo genere:

$$I = \{A_0, A_1 ... A_n\} := \{A_0\} \cup \{A_1\} \cup ... \cup \{A_n\}$$

Definendo ogni insieme con più elementi tramite l'unione dei singoletti dei suoi elementi.

2.2.7 Definizione dei numeri naturali e caratteristiche

Definiamo ogni codifica del numero naturale partendo da $\emptyset = 0$ e definendo in modo ricorsivo

$$[[N+1]] = [[N]] \bigcup \{[[N]]\}$$

Usiamo la notazione $[[\ ]]$ per le definizioni, una relazione, una implementazione di un concetto astratto

2.2.8 Assioma dell'infinito

Questo assioma permette la creazione di un insieme infinito da cui poi si possono **creare i numeri, in finiti**, unito all'assioma potenza si possono creare infiniti ancora più grandi, dato che possiede all'interno tutte le codifiche dei numeri naturali, qualcosa di *metamatematico*. Alcuni matematici pensano che sia una classe.

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

Definisce in modo univoco ogni elemento di N, partendo dagli insiemi, esiste una biunivicità fra elementi di questo insieme e elementi dei numeri naturali.

Abusi di notazione

L'insieme infinito così definito si indica con N

2.2.9 Assioma dell'insieme potenza

Questo insieme è utile per **espandere l'infinito** ossia creare degli insiemi ancora più grandi.

È il primo tra gli assiomi per ora esistenti che può avere qualcosa di controverso.

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

Ossia l'insieme Y contiene tutti gli insiemi di X.

Abusi di notazione

Possiede due abusi di notazione possibili: come
$2^{\{1,2\}}$ or $P(x)$

2.2.10 Assioma di regolarità o fondazione

$$\forall A(A \neq \emptyset \implies \exists B(B \in A \wedge \nexists C(C \in A \wedge C \in B)))$$

> Ogni insieme non vuoto ha un elemento dal quale è disgiunto.

Fra le conseguenze: **nessun insieme contiene (ricorsivamente) se stesso** e ha quindi senso cercare di misurare la taglia (chiamata cardinalità) di un insieme.

Si chiama di fondazione perché **evita la ricorsione infinita** permettendo, quindi, una fine. Sarà in seguito su questa fine che si baserà il concetto di cardinalità di un insieme.

Ossia l'insieme A deve possedere per questo assioma un elemento per cui l'intersezione sia vuota).

2.2.11 Assioma di rimpiazzamento

Questo è un assioma dibattuto, in pratica mi dice che posso costruire un altro insieme a partire da un insieme e una funzione.

$$\forall B(\exists C(P(B,C)) \wedge (P(B,C) \wedge P(B,C') \rightarrow C = C')) \rightarrow \forall A(\exists D(\forall C(C \in D \iff \exists B(B \in A \wedge P(B,C)))))$$

> Intuitivamente: l’immagine di un insieme rispetto a una formula che descrive una funzione è ancora un insieme.

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.

▼ Intuizione

Data una funzione che associa elementi fra due insiemi, allora questo assioma stabilisce che l'insieme d'arrivo, l'immagine, esiste come insieme (cioè è un insieme ben definito

Questo assioma è **necessario per gli infiniti** e gestire queste cose, per le cose finite non serve.`

2.3 Regole di dimostrazione

Queste regole sono presentate con maggiore rigore in Deduzione naturale

2.3.1 Regole di introduzione e eliminazione

Eliminazione mi serve per restringere sull'ipotesi, un risultato intermedio per avere qualcosa che voglio.

Questa eliminazione può essere utilizzata con $\forall X.P(X)|| \implies$

Introduzione

$\subseteq$

2.3.2 Abbreviazioni

sia x tale che P(X) significa che prendo ogni x tale che valga quello. Probabilmente per dimostrare un certo risultato Q(X).

Di solito si fa una lunga catena di relazioni da ipotesi che finiscono con un quindi per asserire la tesi.

2.3.3 Esempi di dimostrazione

Riflessività di $\subseteq$, Asimmetria di $\subseteq$, transività di $\subseteq$

2.3.4 Dimostrazione per assurdo

Si utilizzano le ipotesi per dimostrare una contraddizione, per cui l'ipotesi è falsa.

Posso concludere qualunque cosa dopo la dimostrazione per assurdo **ex-falso quodlibet** abbiamo solamente dimostrato che le ipotesi sono false (quindi se le ipotesi sono binarie, è vero l'opposto).

Significa che una volta giunto ad un assurdo posso dire qualunque cosa, in particolare quello che mi serviva, terminando la dimostrazione.

L'assurdo si ha quando si ha una antinomia, un pò come un paradosso in cui si conclude P e Not P.

NON P

si può definire non P come P $\implies$ Assurdo

Poniamo che $P = \text{true}$ allora $\bar{P} \implies \text{absurd}$ passando da qualche ragionamento, cosa che chiaramente è contro la tesi.

Dimostrazione di ex-falso quodlibet

IP: $P, \bar{P}$

HP: B

essendo vera P, è vera l'unione $P \cup B$ partendo da questa ipotesi, e unendola con l'ipotesi $\bar{P}$ si sa che P è falso allora B deve essere vera affinché $P \cup B$ sia vera. Ecco che da un assurdo si ha qualunque cosa.

2.4 Relazioni fra insiemi

Vedere la pagina di appunti sulle relazioni fra insiemi (classi di equivalenza e simili, con anche le funzioni!) Relazioni fra insiemi

3 Relazioni fra insiemi

3.1 Coppia ordinata

3.1.1 Definizione di Kuratowsky

Una coppia ordinata è definita dall'insieme

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

È quindi chiaro che due coppie ordinate sono uguali fra di loro nel caso in cui gli elementi sono uguali ma anche la loro posizione sono uguali

▼ Teorema caratterizzazione delle coppie

Teorema di caratterizzazione delle coppie

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

Dimostrazione: omessa

3.1.2 Definizione di Wiener

$$(X,Y) := \{\{\{X\},\emptyset\},\{\{Y\}\}\}$$

3.1.3 Definizione di Hausdorff

$$(X,Y) := \{\{X,1\},\{X,2\}\}$$

3.1.4 Proprietà fondamentale coppie ordinate

Due coppie ordinate si dicono uguali se e solo se il primo elemento dei due sono uguali e la stessa cosa per il secondo

▼ Dimostrazione

Proposition 1.2.1 *If (a,b) and (c,d) are ordered pairs and $(a,b) = (c,d)$, then $a = c$ and $b = d$.*

Proof The proof makes repeated use of the extension axiom. First, suppose that $a = b$. Then $(a,b) = \{\{a\}\} = \{\{c\},\{c,d\}\}$, and so $\{c,d\} = \{a\}$, and $a = c = d$. Thus $a = b = c = d$. Similarly, if $c = d$ then $a = b = c = d$.

Finally, suppose that $a \neq b$ and $c \neq d$. Since $\{a\} \in (c,d)$, either $\{a\} = \{c\}$ or $\{a\} = \{c,d\}$. But if $\{a\} = \{c,d\}$ then $c = a = d$, giving a contradiction. Thus $\{a\} = \{c\}$ and $a = c$. Since $\{a,b\} \in (c,d)$, either $\{a,b\} = \{c\}$ or $\{a,b\} = \{c,d\}$. But if $\{a,b\} = \{c\}$, then $a = c = b$, giving a contradiction. Thus $\{a,b\} = \{c,d\}$, and so $b = c$ or $b = d$. But if $b = c$ then $b = c = a$, giving a contradiction. Thus $b = d$. □

If A is a set, then all its members are sets, and they, in turn, can have members.

▼ DImostrazione di wiki, difficile

Kuratowski:
If. If $a = c$ and $b = d$, then $\{\{a\},\{a,b\}\} = \{\{c\},\{c,d\}\}$. Thus $(a,b)_K = (c,d)_K$.

Only if. Two cases: $a = b$, and $a \neq b$.

If $a = b$:

$(a,b)_K = \{\{a\},\{a,b\}\} = \{\{a\},\{a,a\}\} = \{\{a\}\}$.
$(c,d)_K = \{\{c\},\{c,d\}\} = \{\{a\}\}$.
Thus $\{c\} = \{c,d\} = \{a\}$, which implies $a = c$ and $a = d$. By hypothesis, $a = b$. Hence $b = d$.

If $a \neq b$, then $(a,b)_K = (c,d)_K$ implies $\{\{a\},\{a,b\}\} = \{\{c\},\{c,d\}\}$.

Suppose $\{c,d\} = \{a\}$. Then $c = d = a$, and so $\{\{c\},\{c,d\}\} = \{\{a\},\{a,a\}\} = \{\{a\},\{a\}\} = \{\{a\}\}$. But then $\{\{a\},\{a,b\}\}$ would also equal $\{\{a\}\}$, so that $b = a$ which contradicts $a \neq b$.

Suppose $\{c\} = \{a,b\}$. Then $a = b = c$, which also contradicts $a \neq b$.

Therefore $\{c\} = \{a\}$, so that $c = a$ and $\{c,d\} = \{a,b\}$.

If $d = a$ were true, then $\{c,d\} = \{a,a\} = \{a\} \neq \{a,b\}$, a contradiction. Thus $d = b$ is the case, so that $a = c$ and $b = d$.

3.2 Prodotto cartesiano

3.2.1 Definizione del prodotto

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

Utilizzando un linguaggio naturale, stiamo prendendo tutti gli elementi da due insiemi e stiamo prendendo una coppia ordinata: *prendiamo tutte le coppie ordinate possibili*.

questo si indica con $A \times B$

3.2.2 Relazione e con il Vuoto

> Una relazione è una **qualunque sottoinsieme** di $A \times B$

Questa cosa si scrive come $a \mathcal{R} b \iff \langle a, b \rangle \in \mathcal{R}$

Teorema relazioni da e verso insiemi vuoti.

Se $\mathcal{R} \subseteq A \times \emptyset \text{ or } \emptyset \times A \text{ allora } \mathcal{R} = \emptyset$ questo si dimostra che il prodotto cartesiano con un insieme vuoto è sempre vuoto, quindi l'unica relazione esistente è il vuoto.

3.3 Funzione

Per ogni elemento di un elemento dominio, si ha solo una unica immagine in un insieme d'arrvo immagine, si scrive $X f Y$:

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

L'abuso di notazione tipico dei matematici è una falsità perché sembra che la funzione calcoli Y, inverità non calcola niente, ma solamente è una relazione. Ecco l'abuso di notazione.

3.3.1 Spazio di funzioni

$\forall A, \forall B, \exists C, \forall f (f \in C \iff$
$\text{f è una funzione dal dominio A e codominio B e si ha che C} = B^A)$

3.3.2 Funzioni da e verso insieme vuoti

Se è il codominio vuoto, allora non esistono funzioni possibili perché non esistono relazioni possibili, non ho nessun elemento da collegare agli elementi del dominio.

$\emptyset^A = \emptyset$

Se il dominio è vuoto, allora esiste la funzione vuota, che non fa nulla, perché tanto non ho nessun X appartenente a A da loopare, non faccio niente quindi creo l'insieme che non ha nulla.

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

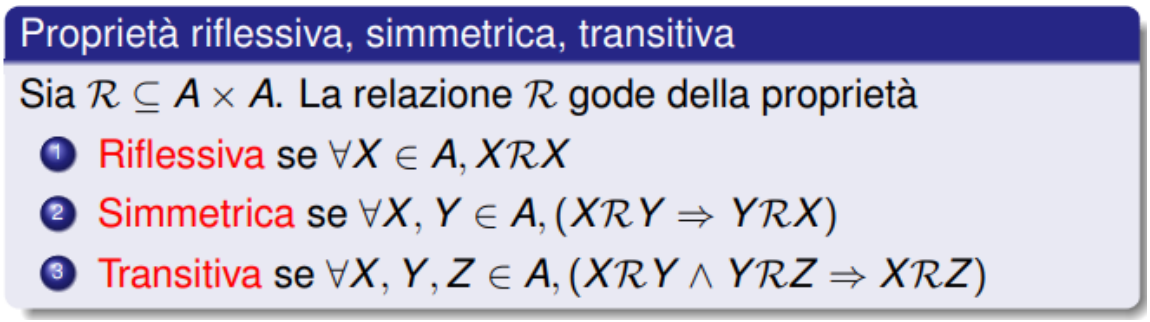
Se sia codominio che dominio sono vuoti allora devo prima loopare nel dominio, che è vuoto, quindi già l'insieme vuoto ho creato.

3.4 Relazioni RST

3.4.1 Proprietà delle relazioni

- Riflessiva se $\forall X, X\mathcal{R}X$
- SImmetrica $\forall X, \forall Y\, X\mathcal{R}Y \implies Y\mathcal{R}X$
- Transitiva $\forall X, Y, Z(X\mathcal{R}Y \land Y\mathcal{R}Z \implies X\mathcal{R}Z)$

▼ Proprietà



▼ Esempi

= Vale tutti e tre

< Transitiva non simmetrica e non riflessiva

≤ transitiva e riflessiva ma non simmetrica

≠ è simmetrica e bbasta

3.4.2 Ordinamento stretto

Una funzione che sia transitiva e non riflesssiva, per esempio il < o il >

3.4.3 Ordinamento lasco

Una funzione che sia transitiva e riflessiva per esempio ≤ o il ≥

In più si può dire che sia antisimmetrica, cioè che se vale $x\mathcal{R}y \land y\mathcal{R}x \implies x = y$

Un altro buon esempio è la relazione di divisione.

3.4.4 Equivalenza

Se ha tutte e tre le proprietà si può dire che sia una relazione di equivalenza.

Questa relazione è **utile per confrontare oggetti** perché è come dire che sono la stessa cosa due elementi quando soddisfano una relazione di equivalenza.

Dire che sono uguali è stato un qualcosa di cui la matematica si è interessata storicamente, dire uguale è diverso da dire che sono equivalenti.

Esercizio

▼ Difficile

Teorema: per ogni f,g ∈ {x}^A (spazio delle funzioni di codominio {x} e dominio A), f ⊆ g.

(Simboli utilizzabili: ∈ ∅ ∪ ∩ ⊆ ∀ ∃ ∧ ∨ ¬ ⇒ ⇔)

Suggerimenti: serve utilizzare la definizione di funzione e sostituire oggetti con altri uguali nella prova (ove l'uguaglianza è una ipotesi o un precedente risultato intermedio).

Soluzione

Dimostrazione: siano A,x,f,g t.c. f ∈ {x}^A (H1) e g ∈ {x}^A (H2). Dobbiamo dimostrare f ⊆ g, ovvero ∀W. W ∈ f ⇒ W ∈ g. Sia W t.c. W ∈ f (K1). Per H1 e l'assioma dello spazio di funzioni, f è una funzione di dominio A e codominio {x}. Per definizione di funzione, f ⊆ A × {x}, ovvero ∀C. C ∈ f ⇒ C ∈ A × {x}. Quindi, per K1, W ∈ A × {x}. Per il teorema del prodotto cartesiano ∃a,b. W = ⟨a,b⟩ ∧ a ∈ A ∧ b ∈ {x}. Siano a, b insiemi t.c. W = ⟨a,b⟩ (K2) e a ∈ A (K3) e b ∈ {x} (K4). Per K4 e l'assioma del singoletto, b = x (K5). Dobbiamo dimostrare W ∈ g. Per K2 possiamo ridurci a dimostrare ⟨a,b⟩ ∈ g. Per la definizione di funzione, ∀z. z ∈ A ⇒ ∃!y. y ∈ {x} ∧ ⟨z,y⟩ ∈ g. Quindi, per K3, ∃!y. y ∈ {x} ∧ ⟨a,y⟩ ∈ g. Sia y t.c. y ∈ {x} (I1) e ⟨a,y⟩ ∈ g (I2). Da I1, per l'assioma del singoletto y = x. Quindi, per K5, y = b. Quindi, per I2, ⟨a,b⟩ ∈ g.

qed.

▼ Misc

- $\forall A,\ \forall f, g \in A \times A$ se f, g suriettive allora anche h(x) = f(g(x)) lo è
- Sia c(X) = $\{Y \in U | \neg(Y \in X)\}$ (complementare di X risp a U), se $X \subseteq Y$ allora $X \cap$ c(Y) = $\varnothing$
- $\forall \equiv$ su $A, A \subseteq \cup A/_{\equiv}$

3.5 Classi

$\equiv \subseteq A \times$
A è una relazione di equivalenza, allora la classe di equivalenza di $x \in A$ rispetto a $\equiv$ è definito come $[x]_{\equiv} =^{def} \{y \in A | y \equiv x\}$

3.5.1 Relazioni fra classi di equivalenza

Fra tutte le classi di equivalenza di ha che ho le due classi sono equivalenti fra di loro, oppure sono diverse (disgiunte) fra di loro.

Abbozzo di dim

Preso classe equivalente X, Y,allora per la transività Z è transitivo sia a Z X sia a Y, e poi si può utilizzare, usiamo l'assioma di estensionalità per dimostrare che le due classi sono uguali.

Poi cerco l'intersezione, se nell'intersezione di due classi di equivalenza trovo un Z, ho che queste due classi sono identitiche, e se sono identiche so che ci sono Z e simili.

3.5.2 Insieme quoziente

L'insieme quoziente **contiene tutti gli elementi** (classi di equivalenza) possibili (disgiunti per la classe di equivalenza).

è definita come, fatto con l'assioma di rimpiazzamento (posso creare un insieme se possiedo una funzione)

$$U_{/\equiv} := \{[x]_{\equiv} \mid x \in U\}$$

Utilità

Questi insiemi sono utili per costruire ancora, scegliere qualche proprietà a seconda del bisogno.

3.5.3 Costruzione di $\mathbb{Z}$

Partendo dall'insieme del prodotto cartesiano $\mathbb{N} \times \mathbb{N}$ definiamo ogni coppia $\langle a, b \rangle$ come $a - b$. Allora possiamo definire una classe di equivalenza per il risultato di una sottrazione...

Preso l'insieme quoziente di tutte queste classi di equivalenza si può creare $\mathbb{Z}$ e lo possiamo indicare con numeri come al solito.

Quindi invece di indicare un numero in Zeta come una classe di equivalenza, indico normalmente con + -.

3.5.4 Costruzione di $\mathbb{Q}$

Uguale a Z solo che invece della sottrazione creo la somma

Esistenza funzione bigettiva N → Q (1)

3.6 Cardinalità di un insieme

3.6.1 Intuizione dalle funzioni

Si può creare una prima intuizione dal concetto di iniettività, suriettività e bigezione di funzioni fra due insiemi sul concetto di cardinalità

Iniettività

Se il codominio fosse più piccolo del dominio, per il principio dei cassetti deve esserci una relazione che punta allo stesso elemento nel codominio, per cui la cardinalità del codominio deve essere più grande

Suriettività

Se il dominio fosse più piccolo del cominio, non avrei abbastanza frecce per raggiungere tutti gli elementi del codominio, quindi sarebbe impossibile una funzione suriettiva.

Bigettività

Se per iniettività e suriettività i due insiemi devono essere uguali per cardinalità

3.6.2 Definizione di cardinalità

Due elementi hanno la stessa cardinalità **sse esiste una bigezione fra i due**, e quindi possono definire una classe di equivalenza indicata

$U_{/\equiv}$ e posso poi definire anche una classe quoziente delle relazioni di equivalenza.

3.6.3 Definizione con insiemi

È possibile, con un lunghissimo lavoro, costruire questa classe attraverso solamente gli insiemi questa classe, invece di utilizzare le classi di equivalenza.

In altre parole si può dimostrare che è abbastanza piccola la classe dei numeri cardinali

Critica al matematico

Il matematico indica con lo stesso numero un numero cardinale e il numero naturale, ma per definizione di numero cardinale e numero naturale sono diverse, il primo è una classe di quivalenza ddell'insieme {1,2,3} (che contiene in sé tutti gli insiemi di 3 elementi, fra qui anche il numero naturale 3, mentre per definizione del numero natuale 3 è {0,1,2}

3.6.4 Abuso di notazione e aleph

Si indica la classe di equivalenza per la classe di equivalenza $[x]_{/\equiv}$ come $|x|$

In particolare per indicare la cardinalità dei numeri naturali è $\aleph_0$

3.6.5 Insiemi infiniti

Fra questi definiamo anche **l'insieme finito** che praticamente è definito come la negazione dell'insieme finito.

3.7 Albergo di Hilbert

Hilbert è stato uno dei matematici più famosi a fine secolo scorso e creò i problemi del millennio per lo sviluppo della matematica attuale.

3.7.1 Albergo finito e infinito

Se è finito allora non si può accomodare in nessun modo.

Ma se invece è infinito? Una soluzione potrebbe essere che ogni cliente si muova nella stanza col numero seguente e si potrebbe trovare di nuovo altro spazio.

3.7.2 Definizione di infinito

> **Infinito** è quando in bigezione con un suo sottoinsieme proprio, ma non è sé stesso.
> In simboli: $A \subset B \land \exists f : B f A$ e f sia bigettiva.

qui infatti esiste un **paradosso**, in quanto essendo essendo un sottoinsieme allora si può dire che sia più piccolo, ma con l'intuizione dell'infinito possiamo dire che hanno la stessa cardinalità, hanno la stessa grandezza.

3.7.3 Ordinamento sugli infiniti <, ≤

≤ Ordinamento lasco

Esiste una classe di equivalenza di ordinamento lasco sse dati due insiemi A, B si ha |A| ≤ |B| se esiste una iniezione fra |A| o |B|

< Ordinamento stretto

È simile al precedente, ma devo togliere l'uguale quindi dico che non esiste una bigezione.

3.8 Diagonalizzazione di Cantor

3.8.1 Dimostrazione

Teorema $|T| < |2^T|$

Dimostrare per assurdo che non esiste una funzione bigettiva da T a 2^T e poi dimostrare che esiste una funzione iniettiva da T a 2^T.

La funzione iniettiva è semplice perché basta mappare ogni T al suo singoletto equivalente in 2^T

In seguito dimostriamo per **assurdo che** non esiste una funzione bigettiva.

Supponiamo una funzione bigettiva, al fine di creare l'assurdo abbiamo bisogno dei tre elementi presentati in Logica meta-linguistica. Quindi meta linguistica, riflessione e negazione.

Definiamo quindi un insieme $A = \{x \in T | x \notin g(x)\}$ data la funzione $g(x)$ bigettiva.

(Possiamo definire x che appartiene all'insieme imamgine perché l'insieme di arrivo sono degli insiemi, in quanto è l'insieme delle parti).

Ma allora data l'iniettività della funzione $g(y) \in 2^T$ esiste un $y \in T$, ma allora $y \in g(x) \iff y \notin g(x)$ e quindi porta all'assurdo.

3.8.2 Sintesi Dim

Data la dimostrazione abbastanza complicata (per me boh) provo a rilistare i passaggi principali utili per questa dimostrazione.

1. Utilizzare l'assurdo per dimostrare l'inesistenza di una funzione bigettiva
2. Utilizzare la suriettività della funzione bigettiva per creare un insieme che possa dare un assurdo.
    a. Allora $g(y) = A$ definito con quella proprietà per assurdo, questa deve esistere per suriettività
3. Devo creare un y appartenente a T l'insieme iniziale perché così comincio a creare qualcosa
4. Dico assurdo perché entrambi i casi, che $y \in A, y \notin A$ creano assurdo perché implicano tra di loro.

3.8.3 |T| < |T^T|

Questo è un corollario della diagonalizzazione di Cantor, che utilizza una semplice disuguaglianza di una funzione caratteristica.

L'unica cosa nuova è $\mathbb{B}^T \leq |T^T|$ questo è vero perché le funzioni che restituiscono un booleano, sono un sottoinsieme delle funzioni che restituiscono T, quindi iniezione è semplice da trovare e si dimostra.

3.8.4 Funzione caratteristica

Data un'insieme booleano, prendiamo un insieme che restituisce vero se l'elemento appartiene, falso se non lo fa.

$\mathbb{B}$ è un insieme con due elementi indicati con 1, 0.

$\chi_c \in \mathbb{B}^A$, posso dire che esiste una bigezione fra questo e l'insieme delle parti di A.

AAAAAA, ovvio, per ogni sottoinsieme C, esiste una unica funzione che mi dice se questi elementi appartengono o meno ad A!

3.8.5 Impossibilità eguaglianza in matematica

Questo teorema ci dice che non si può creare totalmente una funzione implementata che sia precisa come una funzione matematica.

> Data una funzione che va da un insieme grande da un insieme grande, è possibile che non possa essere implementata

▼ Curiosità

Per il matematico, data una qualunque funzione, la probabilità che sia implementabile, è molto vicina a Zero

3.9 Costruzione di \R

Immaginando un numero reale, si ha che un $n \in \mathbb{R}$ in può rappresentare come una parte intera e una sequenza infinita di numeri dopo (di cui possiamo solo approssimare).

3.9.1 Cardinalità dei reali superiore di Aleph 0

Per semplicità, prendiamo la rappresentazione in base due di questo numero, allora questo $\in B^{\mathbb{N}}$

(Questo si può vedere senza molti problemi, quanto la sequenza è infinita, prendo per ogni posizione dopo la virgola arriva a un numero Booleano)

Es. 0,111100001111...

N 0 0123456789....

Ma se queste funzioni appartengono a questo spazio di funzioni, si può finire dicendo che è $|B^{\mathbb{N}}| > |N|$

3.9.2 Esempio sulla densità di R

I numeri hanno possibilmente infinite cifre dopo la virgola, ma è possibile tenerli solamente per $\frac{1}{10^n}$ con n il numero di cifre dopo la virgola (e ci stanno un sacco di numeri).

Questa successione tende chiaramente a 0. Quindi le probabilità sono quasi nulle.

4 Sintassi

> Programmare e dimostrare sono sostanzialmente la stessa attività ~*Coen*

Ma non secondo l'industria...

4.1 Introduzione

4.1.1 Definizione e necessità

> Branca della linguistica, studia creazione di proposizione e il loro collegamento per la creazione di un periodo

In seguito la semantica dà un metodo a queste proposizioni in modo che abbiano un senso.

1. Utile o necessario per la definizione del linguaggio artificiale

4.1.2 Alfabeto, stringa, linguaggio e grammatica

Alfabeto: Insieme non vuoto di simboli (che spesso sono diversi fra di loro)

Stringa seguenza finita (vuoto è possibile) di simboli $\epsilon = \emptyset$

Linguaggio: insieme di stringhe (di qualunque tipo, finito o infinito).

Grammatica formalismo (un insieme di regole che lo rende finito) che definisce un linguaggio

4.2 Backus-Naur Form

Indicato con BNF

4.2.1 Perché BNF

Una formalizzazione informatica che permetta l'elaborazione di grammatiche → notazione per descrivere grammatiche

Non è l'unica ma per gli informatici è la migliore.

4.2.2 Caratteristiche

Indichiamo con (T, NT, X, P) rispettivamente

T = **l'alfabeto**, l'insieme di simboli che usiamo

NT è un insieme di simboli diversi da T (insieme **non terminale**) Sono solamente ausigliari.

X = qualunque elemento di NT, basta che sia **iniziale**

P = simile alla grammatica, sono delle **coppie come produzioni:** comprendono:

1. Non terminali
2. Insieme di stringhe che contengono un pò di tutto ,indicate con ω_n

Es. $(X, \{\omega_1 ... \omega_n\})$ è una produzione.

Quindi questi quattro elementi riescono ad identificare in maniera univoca la semantica di un linguaggio.

Capiremo il senso di questa definizione per l'informatica fra poco.

Indicazione

Si può indicare con $X ::= 0|0Y$ e simili, utilizzando solo per produzioni come coppie è sufficiente per definire una sintassi BNF.

4.2.3 Definizione di un linguaggio

Di solito fra tutte è sufficiente prendere le produzioni per dire un linguaggio.(ha senso supponendo che tutti i simboli della grammatica siano utilizzati)

▼ Immagine di definizione

La stringa ω di soli terminali appartiene al linguaggio sse ottengo ω a partire da X rimpiazzando ripetutamente ciascun non terminale con una delle stringhe alternative a lui associate in una produzione.

Processo iterativo che parte dal non terminale e arriva a stringhe finite.

▼ Esercizio

Dimostrare che 000 non appartiene a questo linguaggio

Esempio di prima:
$X ::= 0 \mid 0Y$
$Y ::= 1X$

01010 appartiene al linguaggio poichè
$X \to 0Y \to 01X \to 010Y \to 0101X \to 01010$
000 non appartiene al linguaggio

4.3 Ambiguità in BNF

4.3.1 Definizione ambiguità

Se si può definire in due modi diversi la stessa parola, allora si dice che il **linguaggio è ambiguo**.

4.3.2 Soluzione ambiguità (3)

Queste non sono sempre necessarie in ogni grammatica, ma sarebbero utili per la comprensione

1. Ordine di precedenza per operatori
2. Associatività per ogni operatore (cioè se l'operatore prende solo a destra o da sinistra)
3. Parentesi?

Così **definisco un ordine di precedenza** quindi risolvo le ambiguità, ma non dovrei fare in questo modo.

4.4 Albero di Sintassi astratta

Buona cosa potrebbe essere la pagina di wiki.

Afferma che questo albero è molto utile per il compilatore (così capisce cosa stiamo provando a fare).

Un approfondimento possibile per questi alberi è la Contex-free-grammar ovvero come evitare l'ambiguità del linguaggio naturale. (simile a BNF).

4.4.1 Definizione (4)

Prima si deve trovare una **BNF non ambigua**, poi possiamo creare un albero di questo genere.

▼ Definizione di albero di sintassi

> L'AST ha un nodo per ogni espansione di simbolo. La radice corrisponde al simbolo iniziale e le foglie alle espansioni fatte con solo simboli terminali. I nodi figli di un nodo corrispondono ai non terminali contenuti nell'espansione del nodo padre.

Di solito ogni linguaggio di programmazione è prima trasformato in un albero di sintassi e in seguito il compilatore elabora su questa cosa.

4.4.2 Ricorsività: le sottoformule immediate

Sono figli diretti, sottoformule immediate generate dal nodo padre.

Ecco una struttura di dati ricorsiva, impareremo a sfruttare questa caratteristica della ricorsione.

Questa sottostruttura ricorsiva è molto utile perché so anche come è stato ricavato, non solo so se appartiene o meno! Più informazioni! Riusciamo ad assegnare un significato.

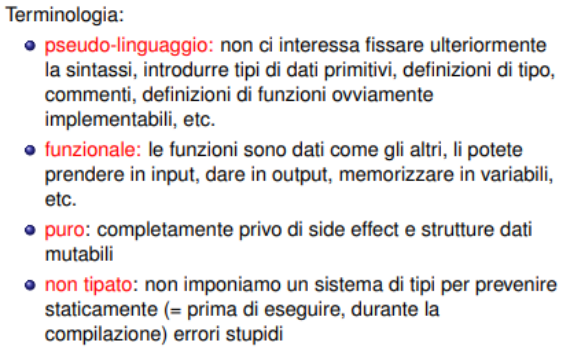
La cosa bella di questra struttura è che possiamo utillizzare la stessa funzione (programma o quel che si voglia chiamarlo) per risolvere il problema su alcuni dati più piccoli (sotto dati).

4.5 Pseudo-linguaggio funzionale puro non tipato

Altolivello- vicino dominio del problema, senza alcuni dettagli di implementazione (che farà da solo).

Basso livello- vicino al dominio della soluzione ossia tratta alivello vicino al computer.

Tutti questi linguaggi hanno un albero sotto, che cerca di utilizzare questa grammatica formale per capire ciò che è scritto.


Terminologia:

- pseudo-linguaggio: non ci interessa fissare ulteriormente la sintassi, introdurre tipi di dati primitivi, definizioni di tipo, commenti, definizioni di funzioni ovviamente implementabili, etc.
- funzionale: le funzioni sono dati come gli altri, li potete prendere in input, dare in output, memorizzare in variabili, etc.
- puro: completamente privo di side effect e strutture dati mutabili
- non tipato: non imponiamo un sistema di tipi per prevenire staticamente (= prima di eseguire, durante la compilazione) errori stupidi

4.5.1 Significato del nome

Pseudo linguaggio perché questo linguaggio non esiste realmente, è solamente qualcosa di simile, di vicino al un linguaggio reale (meno sintassi diciamo)

funzionale si utilizzano funzioni, sia come input, output per memorizzare cose e simili

Puro senza side effect, senza storare variabili e fare cicli while o for

Non tipato senza che un compilatore si lamenti di come è implementato il tipo, quindi maggiore astrazione anche da questo punto di vista

4.5.2 Funzioni unarie (3)

Le funzioni unarie sono definite da tre parti principali:

1. Il nome della funzione
2. Un pattern ω, di solito una stringa dell'alfabeto
    a. Variabili $\rightarrow$ Non terminali
    b. Costruttori e simili $\rightarrow$ terminali costruiti con la gramatica del linguaggio.
3. Corpo, quello che è dentro la funzione
    a. Chiamate ad altre funzioni
    b. Parametri formali e altre costanti
    c. Condizioni di control flow

4.5.3 Pattern matching

Questa è la definizione di matching

ρ fa match con ω se ottengo ρ da ω sostituendo a ogni non terminale $X_i \in \omega$ una sottostringa ρ_i di ρ.

In modo intuitivo: Match = se p terminale matcha ω se partendo da ω si può creare p

Chiamate di funzione

In questo linguaggio funzionale, andremo ad utilizzare Haskell, la chiamata di funzione avviene per pattern match.

Passo a sostituire i parametri formali a seconda di cosa matchi, ricostruendo tutto continuando.

4.5.4 Side effects

Questi linguaggi funzionali non devono avere side effects, ossia non devono accedere a locazioni di memoria fuori dal loro scope, o fuori dai propri parametri formali, quindi molto più controllabile.

Ma questo significa che non abbiamo la libertà di allocare memoria e simili.

4.5.5 Potenza espressiva

Noi non possiamo programmare tutto quello che la matematica può fare.

Ma certe cose si possono fare in un linguaggio e non in un altro. Ma qualunque funzione in qualunque altro linguaggio potrebbe essere espresso nello pseudo-codice attuale (quando questa cosa accade **Turing-completezza**).

Ma dato che non abbiamo side effect non ci interessa I/O e video o simili.

4.6 Ricorsione strutturale

Una funzione $f(\omega)$ dove ω è una stringa (formula) è definita per ricorsione strutturale sse

1. f considera tutte le possibili produzioni che definiscono ω una e una volta sola
2. per ogni produzione f si richiama ricorsivamente solamente sulle sottoformule immediate di ω

4.6.1 Solito ragionamento per ricorsione

Come di solito le ricorsioni, se è un caso base allora risolvo subito, in modo diretto.

Altrimenti risolvo ricorsivamente un sotto problema più facile, ma è ancora lo stesso problema, ecco perché strutturale → Hanno la stessa struttura, quindi sto utilizzando la stessa funzione per risolvere lo stesso problema ma per input diversi.

Quindi risolvo problemi più piccoli e poi le ricompongo alla maniera iniziale.

1. Strutture uguali (cioè i sottodati devono essere ancora dei tipi dei dati iniziali, se ho in input lista di qualcosa e poi ho totalmente altro non posso fare).
2. Risolvo così problemi più semplici in modo ricorsivo.

4.6.2 Errori comuni

DI solito la ricorsione è difficile perché le persone tendono a cercare di scoprire in che modo sia implementata la ricorsione, cioè cercano di comprendere cosa faccia la ricorsione a livello troppo basso.

- Chiamate ricorsive non sui sottoproblemi
- Struttura della ricorsione è errata (usando produzioni inesistenti)
- Mancare di qualche produzione

4.6.3 Esercizi Ricorsione strutturale

Questi esercizi sono importanti dato che poi all'esame dovrai risolvere qualcosa di simile!

▼ Es 1

```
-- Problema 1: data una lista (di numeri)
-- calcolare l'insieme potenza della lista
```

▼ Es 2

```
-- Problema 1: data una lista (di numeri)
-- calcolare la lista di tutte le permutazioni
-- della lista in input
-- Es: dato 1:2:3:[], restituire
-- (1:2:3:[]):(1:3:2:[]):(2:1:3:[]):(2:3:1:[]):
--  (3:1:2:[]):(3:2:1:[]):[]
-- Soluzione: per l1
```

▼ Es 3

Uno tostino come esercizio è definire una funzione che ritorni vero se e solo se un elemento compare due volte nell'insieme.

▼ Es 4

N ::= O | S N

dove il
simbolo terminale O rappresenta lo 0 e il simbolo terminale S, letto "successore", dato un numero naturale N forma il numero naturale S N che segue N nella numerazione.

Esempio: 3 viene rappresentato in base 1 come S (S (S O))) e 5 come S (S (S (S O)))).

Nota:
la rappresentazione corrisponde al modo con cui i bambini imparano a contare, usando le dita. O è il pugno chiuso e ogni S corrispondere ad aggiungere un dito.

Problema 1: definire per ricorsione strutturale una funzione + sui numeri naturali in base 1 che ne implementi la somma

Esempio: S (S O) + S (S (S O))) = S (S (S (S (S O)))))

Suggerimento: procedere per ricorsione strutturale sul primo argomento

Problema 2:

definire per ricorsione strutturale una funzione * sui numeri naturali in base 1 che ne implementi il prodotto

Esempio: S (S O) * S (S O) = S (S (S (S O))))

Suggerimento: per implementare il * potete usare il +

Problema 3:

definire per ricorsione strutturale una funzione ^ sui numeri naturali
in base 1 che elevi il primo numero alla potenza indicato dal secondo

Esempio: S (S O) ^ S (S (S O))) = S (S (S (S (S (S (S O))))))))

Suggerimento: scegliere bene su quale input procedere per ricorsione strutturale

Questi dovrebbero essere difficili, se sai risolvere questi, dovresti essere in grado di farlo per tutti.

4.7 Induzione strutturale

Questa è una **tecnica dimostrativa** per dimostrare che una struttura gode di una certa proprietà.
È strettamente legata alla ricorsione perché la ricorsione è il calcolo della soluzione mentre l'induzione la dimsotrazione della correttezza.

Questa forma di dimostrazione è valida per ragioni molto simili alla ricorsione strutturale, perché ogni passo è giustificato dal precedente, di cui il caso base è assunto come vero.

Quindi bisogna prima capire quali siano le differenze fra induzione e strutturale.

4.7.1 Il procedimento

1. L'output deve essere una dimostrazione
2. Si suppone che valga per tutti i sottocasi di questo di input (in pratica uguale alla ricorsione, per tutti gli sottoinput immediati stiamo supponendo che valga) come in matematica puoi affermare che valga per tutti i numeri minori di n come ***ipotesi induttiva.***

4.8 Confronto funzioni mate e info

Questo paragrafetto si rifà all'iniziale introduzione sui Logica meta-linguistica sui paradossi in matematica e informatica.

4.8.1 Matematica

Rappresentazione è fatta con relazioni, sottoinsieme del prodotto cartesiano. Questa è **inefficiente** dal punto di vista del calcolo in quanto non ci da un modo per creare un calcolo. (non posso scorrere perché le liste restano illimitate).

Si in questo caso stai pensando alle Relazioni fra insiemi non al modo per calcolarle.

4.8.2 Informatica

Di solito gli algoritmi ragionano in modo simile in basi diverse, that is l'efficienza degli algoritmi è molto simile in basi diverse, tranne in base 1 che è esponenzialmente più grande rispetto alle altre basi.

▼ Esempio di def. di funzioni somma

```
O `+ m = m
S n `+ m = S (n `+ m)
-----
n +' ) = n
n +' S m = S (n +' m)
------
O ``+ m = m
S n ``+ m = n ``+ S m
-------
n +" O = m
n +" S m = S n +" m
```

Queste sono quattro procedure di calcolo per la somma non uguali in quanto calcolano diversamente.

Il prof. ha detto (non ho capito il motivo) per cui quelli con un singolo apice utilizzano la stack, mentre invece quelli con due apici utilizzano la heap), non ho capito perché, ma tanto lo spiegherà ad architettura.

4.8.3 Specifiche di funzioni

Possiamo utilizzare le dimostrazioni per induzione strutturale per verificare la correttezza di una funzione.

Per esempio una funzione di concatenzazione dovrebbe soddisfare questi teoremi

Di cui il primo mantiene il numero , il secondo appartenenza, il terzo l'ordine.

1. $|l_1 f l_2| = |l_1| + |l_2|$
2. $x \in l_1 \implies l_1 f l_2$
3. $\forall n, n \leq |l1| \implies$ nth n l1 = nth n (l1@l2) $\wedge \forall n, n \leq |l2| \implies$ nth n l2 = nth (|l+1| +n) (l1@l2)

Se una funzione soddisfa questi teoremi allora possiamo definire in modo rigoroso una funzione.

▼ Funzioni helper per questo

```
cat [] l2 = l2
cat (t:l) l2 = t:cat l l2

length [] = 0
length (n:l) = 1 + length l

-- nth n l tira fuori n-elemento di l
head [] = []
head (n:l) = n

tail [] = []
tail (n:l) = l

nth 0 l = head l
nth (S n) l = nth n (tail l)

nthCorto 0 (t:l) = t
nthCorto (S n) (t:l) = nthCorto n l
```

E poi dovrei verificare ogni singola funzione di questo...

Spesso nella vita reale non c'è bisogno di questo, si scriva specifica parziale

5 Verità e conseguenza Logica

Questa è una necessità per stabilire il significato di una sintassi definiti.

5.1 Verità e Realtà

La verità ha solamente senso quando lo si relaziona con un mondo sensibile, ossia il mondo che si può percepire con i nostri sensi.

5.1.1 Verità parametrica e assoluta

Se un esperimento è ripetibile all'interno del mondo sensibili allora questa è considerata come una **verità parametrica**, ossia dipende da uno stato del mondo sensibile.

verità assoluta come le costanti della fisica.

5.1.2 Scienze pure e scienze molli

La differenza fra queste due è che le scienze pure non stanno parlando del mondo sensibile, ma crea un mondo a sé, astratto, in cui ha senso tutto ciò che viene detto riguardo a questo mondo preciso. Anche l'informatica è teorica.

Ma cosa è la verità senza il mondo sensibile? L'esempio che abbiamo fatto prima non funziona più.

> Le scienze molli descrivono la realtà sensibile mentre le scienze pure descrivono un mondo astratto inesistente nella realtà. (quindi fisica non è pura, secondo questo).

5.1.3 Teoria matematica

È come una storia descritta prima: **insieme di sentenze**,connotazioni.

1. Enti primitivi di un mondo
2. Assiomi che valgono in certi mondi

💡 Gli assiomi sono come le leggi fisiche di un mondo fantastico: descrivono delle regole che valgono in un mondo preciso, e quindi si possono derivare delle conseguenze e le interazioni fra queste.

5.1.4 Modello matematico

Interpretare i concetti primitivi in modo che valgano tutti gli assiomi, questo è il modello.

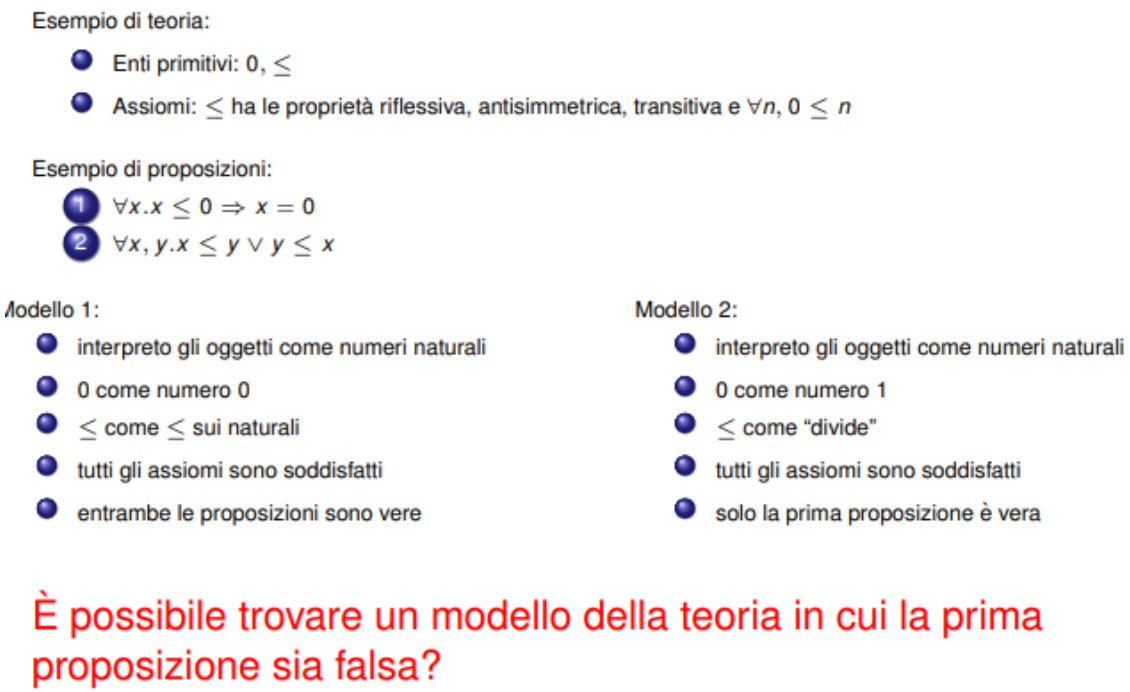
Quindi il modello matematico è una **interpretazione** degli enti primiti e edegli assiomi del mondo!

Non è definito a priori ma si dà il senso

▼ Esempio in classe

Per esempio il mondo con i numeri colorati, quella relazione resta la stessa, posso colorare come mi pare, allora la prima proposizione della slide sono falsi

▼ Esempio teoria e modello



5.2 Il mondo

Il mondo è una **descrizione completa** delle caratteristiche e regole del mondo (quindi oggetti, leggi e simili). Ovviamente questa è solamente una descrizione ipotetica in quanto non possiamo conoscere tutto di un mondo (non conosciamo tutto nemmeno nel mondo in cui viviamo).

Un assioma è valido solamente in alcuni mondi precisi, spesso ci interessa indagare solamente quei mondi. (È utile indagare solamente un mondo in cui quel determinato assioma valga o meno.)

Il concetto di verità **non ha senso come concetto a sé stante** in quanto dipende dal mondo in cui la proposizione è valutata

Le teorie e i modelli matematici hanno senso solamente se interpretati in un mondo particolare. Da soli no. Una proposizione ha un concetto di verità solamente se ha poi senso all'interno del mondo.

5.2.1 Conseguenza logica

> Data una teoria T(un insieme di sentenze) si dice conseguenza logica F di T quando F vale per tutti i modelli di T, ovvero in tutti i mondi in cui valgono le sentenze della teoria.

Due cose:

1. Vera per tutti i modelli T
2. Vera in tutti i mondi in cui le ipotesi G1 G2 etc è vero, ossia tutti gli assiomi sono veri.

▼ Miniriassunto

Con gli assiomi sto filtrando nei mondi in cui questi assiomi siano veri, quindi prendiamo solamente mondi in cui siano veri questi assiomi, con questi assiomi e enti primitivi creiamo un mondo. Allora poi possiamo avere una semantica, un modo di interpretare che è il modello e da qua possiamo avere proposizioni e conseguenze logiche

▼ Detto in altri modi

Gli assiomi creano dei sottomondi in cui esse valgono (sto filtrando sui mondi).

F (un insieme di modelli matematici) è una conseguenza logica di T se vale in tutti i mondi in cui valgono tutte le ipotesi (assiomi) (sentenze) G1 G2 etc... di T

Nota: più ipotesi che ho, più sto restringendo nell'insieme dei mondi, quindi avrò più conseguenze logiche, quindi diventa una cosa più interessante.

5.2.2 Equivalenza logica

Due sentenze sono dette equivalenti se sono soddisfatte esattamente dagli stessi mondi. (ossia filtrano sugli stessi mondi) (assiomi rindondanti uno con l'altro)

▼ Relazione di equivalenza per equivalenza logica

Considera solamente i mondi in cui valgono

Nel secondo caso nella slide ho come ipotesi che entrambi hanno gli stessi mondi cin cui valgono, allora è ovvio che si ha il contrario

In modo simile si ha per il terzo caso

5.3 Valutazione della teoria

Quando è interessante?

Non ha senso chiedersi se un assioma è vero o falso, nemmeno se è giusto o falso, ha solamente senso chiedersi se è vero o falso in un mondo preciso. Vale allora la teoria di consistenza

5.3.1 Inconsistenza di una teoria

> Una teoria è inconsistente quando **non ammette nessun modello**. (ossia non ammette nessuna interpretazione degli enti primitivi in modo che valgano tutti gli assiomi)

Si può anche dire che l'assurdo è conseguenza logica di una teoria inconsistente, quindi è tutto vero, tutto vero per assurdo!?

Ci sono troppi vincoli, quindi sto parlando di niente (ho un insieme vuoto di modelli e quindi sto parlando del vuoto) (tutto è conseguenza logica in questo caso)

Questo vale anche il contrario, però non è dimostrabile in modo semplice anche l'altra freccia.

Quindi se è falso questo, allora ho almeno un mondo in cui tutto ciò che ho è vero! Quindi che sia consistente! (e poi la cosa bella sarebbe cercare le applicazioni pratiche di queste cose)

La Logica studia la conseguenza logica! Introduzione a Logica

5.3.2 Interpretazione

Si può considerare **interpretazione una funzione semantica che per ogni connotazione associa una unica denotazione**, considerato un oggetto di un mondo preciso. (enti primitivi sono connotazioni di un mondo, considerati connotazioni atomiche).

La funzione di interpretazione è descritta meglio in Logica Proposizionale

Le connotazioni sono interpretate come denotazioni del mondo.

Poi ci sono funzioni del mondo e simboli del mondo.

Di solito le connotazioni composte sono ottenute da connotazioni atomiche (denotazioni del mondo) combinate tramite connettivi logici.

5.4 Connotazione denotazione

In informatica sono sintassi e semantica. Sarà ciò che mi serve per evitare l'uso meta-linguistico, invarianza per sostituzione è il primo teorema che serve per questo

5.4.1 Intuizione iniziale

In linguistica la connotazione è il significato psicologico di una parola, mentre la denotazione è la prima cosa che di solito di dà il dizionario, ossia cosa è detto effettivamente.

In breve: (→ indica il significato in informatica)

1. Connotazione: ciò che voglio comunicare, come voglio cominciare (per la logica i messaggi subliminali non sono utili) → **Sintassi** il modo in cui lo sto dicendo
2. Denotazione cosa è detto → **Semantica** ciò che voglio dire, anche considerabile come l'oggetto che viene considerato in questo mondo.

Nel caso dell'informatica l'uso metalinguistico è parlare sulle connotazioni

5.4.2 Teorema Invarianza delle denotazioni

> Data un qualunque contesto (un buco di una frase) e una connotazione, se sostituita con una altra connotazione, le

> denotazioni delle frasi restano le stesse allora quelle due connotazioni sono equivalenti.

Possiamo utilizzare il test di invarianza per vedere se qualcosa è metalinguistico o meno.

Se due connotazioni che possiedono la stessa denotazione hanno output diversi per certi contesti allora questo fa uso metalingusitico

Fondamentale per l'uso metalinguistico

5.4.3 Connettivi logici Intro

Per maggiore precisione sui connettivi logici guardare Connettivi Logici, correttezza, variabili

Abbiamo bisogno di tenere certe cose fisse in modo da tenere un ordine generale fra i mondi. Questi sono i connettivi logici che hanno lo stesso senso ovunque.

Possono essere

Binari

Unari

0-ari

E poi quantificatori come esiste e per ogni

6 Logica proposizionale

Con la logica proposizionale studiamo le denotazioni che hanno un valore di verità, ovvero deve essere una sentenza assertiva. Studio solamente le connotazioni che hanno una capacità denotativa, in quanto è solo quello ch emi importa.

6.1 La sintassi

Vengono qui definite le produzioni che valgono in ogni singolo mondo.

$$F ::= \top | \bot | A | B | ... | \not{F} | F \wedge F | F \vee F | F \implies F$$

Questa è la BNF della nostra sintassi.

Le lettere sono asserzioni, sono sentenze dichiarative a cui posso dare un valore

Connotazioni atomiche

Sono solamente ABCD e le singole lettere che rappresentano le proposizioni.

6.1.1 Formalizzazione

Ovvero il tentativo di esprimere asserzioni attraverso la sintassi della logica proposizionale.

Quindi dare un valore logico a frasi come oggi piove, marco è bello, marco è brutto e simili.

Quest è importante perché è probabile che all'esame ci sia un esercizio di formalizzazione

6.1.2 Note per attenzione

Fare attenzione ai sinonimi e contrari, che devono avere la stessa lettera, non è ovvio.

Una altra cosa su cui fare attenzione sono le connotazioni ovvero modi diversi per dire la stessa denotazione come:

$A \implies B$ è la denotazione delle connotazioni : A è condizione sufficiente di B, B è condizione necessaria di A e altro.

6.2 La semantica

> La semantica classica associa il **valore di verità** per ogni connotazione del linguaggio. Allora si dice che stiamo utilizzando la **logica proposizionale classica**

Noi parliamo tutti i giorni con una semantica naturale, principale o intesa sono tutti dei sinonimi

Devo partire da alcune premesse filosofiche che mi portano alla logica classica.

$filosofia \implies logica \implies matematica$

Nota di wiki

La semantica studia il significato delle parole, quindi ci dice cosa deve significare una connotazione, e possono essere molto diverse: molte semantiche per una stessa sintassi.

▼ Esempio semantica (interpretazione) in prog

Semantica che interpreta la sintassi in termini di tempo:

Interpreto i miei costrutti come il tempo che impiega ad eseguire (molto utile per avere una complessità computazionale (es. if, vado a guardare guardia).

Semantica che interpreta la sintassi in termini di memoria occupata.

6.2.1 Dominio di interpretazione, f semantica

Dominio di interpretazione sono connotazioni (definite tramite BNF)

Dipende dai mondi in Verità, Teorie, modelli, connotazione, denotazione, ossia si può interpretare la connotazione in modi diversi a seconda della denotazione in quello specifico mondo

Si potrebbe dire che ogni mondo possieda una **funzione semantica** che parte da un dominio di interpretazione e denota queste connotazioni con oggetti precisi in un mondo.

6.2.2 Enunciati della logica classica

TODO: fai una parte a sé per questo

Questa logica parla della verità.

Manicheo (visioni estreme)

Sulla falsità e verità

- Ogni enunciato è vero o falso (esistono gradi di verità, secondo alcuni, dibattibile ~Fuzziness)
- Ogni enunciato non può essere vero o falso allo stesso momento

Platoniche

- Staticità il valore di verità non muta nel tempo, immutabile.
    - No libero arbitrio o possibilità di cambiare il proprio futuro (se è una verità...) ah i trip mentali.
- Determinatezza il valore di verità è sempre determinato (simile al primo aristoteo?)
    - Tutto il futuro è predicibile e non si può fare niente (ma mancano le conoscenze).

Utilizzando queste due caratteristiche del valore di verità possiamo già fare delle inferenze:

Differenza fra verità e conoscenza

La conoscenza non possiede il senso di staticità e determinatezza (può cambiare nel tempo (cresce), e non è sempre quella) mentre la verità lo è

Differenza fra verità e risorse

La verità è statica (si può utilizzare quante volte si vuole), mentre le risorse vengono consumate (esempio cibo, se lo mangio scompare,o almeno non è più nella forma attuale).

Critiche della logica intuizionista

Esistono dei mondi non determinati che si possono determinare a seconda dei mondi, non posso avere una verità statica immutabile.

Non vale per la conoscenza e mondi non-deterministici

6.2.3 Booleani e funzione di interpretazione classica

Scegliemo di indicare i booloani come

1 → Vero 0 → Falso

Per tre motivi principali:

1. Posso utilizzare le regole dell'algebra in quanto sono dei numeri e possiedono quelle proprietà
2. Non vanno in confuzione con top e bottom per il loro essere dei numeri (altrimenti avrei dovuto distinguere il vero e il falso a livello sintattivo, a livello denotativo e poi latro)
3. Si può facilmente trovare un intervallo di verità.

Interpretazione

Si può definire (funzione di) interpretazione (classica) di un mondo, possibile solamente grazie all'esistenza di cui sopra dei boleani, che associa a ogni sentenza del mondo un valore di verità.

Allora dato che è una funzione, possiamo notare che possiede i valori di:

1. Staticità, in quanto le funzioni sono solamente quelle e non cambiano
2. Determinatezza per le proprietà delle funzioni, ossia per ogni sentenza del mondo posso associare un valore di verità.

6.2.4 funzione semantica per la logica classica

Questo è definito tramite BNF e fissa il linguaggio della logica in un mondo preciso.

NOTA: distingui bene la differenza fra valore semantico e interpretazione in un mondo.

Definizione: data un'interpretazione (o mondo) v, la semantica $[\![\cdot]\!]^v : \mathcal{F} \to \{0,1\}$, è definita per induzione strutturale sulle connotazioni come segue:

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

6.2.5 Tabella di verità

Le tabelle di verità hanno senso solamente nell'ambito della logica classica, in quanto può **solamente rappresentare una logica finita**,, quindi la classica non la intuizionistica.

Le tabelle di verità sono le stesse viste in [[. (con una nota particolare sulla implicazione materiale, molto diversa dalla normale relazione di causalità utilizzata tutti i giorni

Semantica Tarskiana

È una logica che utilizzano i matematici (molto lapalissiana, inutile dal punto di vista di nuove informazioni che la definizione riesce a dare) (Coen critica anche che questo è perché i matematici se ne fregano della logica, di quello che sta sotto la matematica).

Praticamente utilizza il linguaggio naturale per descrivere la funzione semantica descritta qui sopra, cosa che funziona solamente con la logica classica, e appena si esce da questo reame non ha più senso. Utilizza **troppe nozioni meta-linguistiche**, tanto che potrebbe dire che meta-linquistica e linguistica siano quasi la stessa cosa

6.3 Conseguenza logica (formale)

Questa è la definizione formale di conseguenza logica

$$\Gamma \Vdash F \iff \forall v, (\forall G \in \Gamma, [[G]]^v = 1) \implies [[F]]^v = 1$$

NOTA 1 questa definizione non è computabile, non è direttamente studiabile in ambito informatico, allora c'è bisogno della creazione di un sistema deduttivo. (In questa definizione stiamo parlando di infiniti (di mondi v e assiomi G, che chiaramente non è computabile quindi non è possibile prendere decisioni in questo ambito).

> NOTA 2 Intelligenza artificiale forte, che possa sapere tutto, è ucciso da questa osservazione, in quanto la computazione non può arrivare a sapere tutto. (quindi non può essere un programma... mmmm).

6.3.1 Equivalenza logica

ovvero la semantica per ogni modello v di una teoria è uguale, quindi possiamo dire che valgono per gli stessi mondi. $G \equiv F \iff \forall v, [[F]]^v = [[G]]^v$

6.4 Sistemi deduttivi

6.4.1 Intro

Esistono sistemi deduttivi diversi, quella che vediamo noi è la deduzione naturale.

Possono variare per

- Sintassi diverse
- Esigenze diverse

Cosa è un sistema deduttivo

Il sistema deduttivo è una nozione sintattica che contiene in sé una prova (dimostrazione) di una proposizione. (sintattico perché parla di connotazioni, non di denotazione esatta di quello presente sotto).

Deve quindi utilizzare regole precise per le prove. Ci sono molti modi per fare regole, la più naturale è la deduzione naturale.

6.4.2 Deduzione (3)

$\Gamma \vdash F$, si legge da Γ deduco F, ed è una nuova nozione sintattica (prova esplicita tale per cui la sintassi vada, quasi algoritmico). Poi si potrà dire che ogni dimostrazione deve essere una conseguenza logica e il contrario. Dimostrazione è un dato!

- Deve esserci una **prova esplicita**

  È possibile scrivere una dimostrazione in una qualche sintassi, questo è il significato di deduzione.

Se è sintatizzabile (si può scrivere con una sintassi) **si può trasmettere** ecco che si cerca una sintassi per le dimostrazioni.

- Deve essere **corretta**, ossia $\Gamma \vdash F \implies \Gamma \Vdash F$

  Devo definire il motivo per cui le regole di dimostrazione siano giuste (Le varie regole di introduzione e eliminazione. non vale per le logiche espressive ma solo per la classica!
  Questa nozione è dimostrata in quando parliamo di connettivi, anche se utilizza solamente il concetto di induzione strutturale per dimostrarla...

- **Completezza** (non per logiche espressive (?)) $\forall \Gamma, F, \Gamma \Vdash F \implies \Gamma \vdash F$ vale per logiche semplici, ma esistono dimostrazioni logiche che non saremo mai in grado di dimostrare, in modo simile alla non implementabilità di funzioni matematiche, utilizzando paradossi.

  Questa dimostrazione esiste ma è molto complessa, lo ha saltato di striscio il prof.

6.4.3 La deduzione naturale

Studieremo la deduzione naturale in Deduzione naturale

6.6 Definizioni

6.6.1 Tautologia

$\Vdash F$, quando è è conseguenza logica per tutto. Quindi è anche una **verità assoluta** perché non dipende da mondo in esame. (la tabella logica in esame possiede solamente delgi uno).

Not-tautologia ricorda che il contrario del per ogni è l'esiste il not!, quindi basta un mondo in cui questa proposizione sia falsa.

6.6.2 Soddisfacibilità e non

Soddisfacibilità

Si utilizza lo stesso simbolo $\exists v, v \Vdash F \iff [\![F]\!]^v = 1$ tale che v sia un mondo

Insoddisfacibilità

Quando non esiste nessun mondo, quindi per ogni mondo nno vale la cosa sopra.

(Più o meno questa proprietà è

Taotologicità (teorema)

Si può dimostrare da sopra che una formula è tautologica se per ogni mondo vale che la funzione semantica per input quella proprietà si ha 1

Nota: tautologica implica soddisfacibile, quindi devi fare attenzione!

6.6.3 Conseguenza logica per mondi diversi

Ricondursi al concetto di variabile presente in Connettivi Logici, correttezza, variabili

▼ Ragionamento

Anche $\Gamma \Vdash F$ è rappresentabile con tabelle di verità, ma solo se Γ è un insieme finito (e quindi anche $Var(\Gamma)$ lo è):

1. Sia $n = |Var(F) \cup \bigcup_{G \in \Gamma} Var(G)|$.
   Si costruisce una tabella di verità con 2^n righe (mondi possibili).
2. $\Gamma \Vdash F$ quando F vale 1 in tutte le righe nelle quali tutte le formule in Γ valgono 1
3. In altre parole:
   1. si considerano solamente le righe in cui tutte le formule di Γ valgono 1
   2. $\Gamma \Vdash F$ se F è una tautologia ristretta a tali righe

Nota: quando Γ è vuoto ci si riduce a considerare tutte le righe

Con questo per trovare una conseguenza logica mi posso ridurre a leggere le tabelle di verità.

Drawback

Le righe totali sono 2^n per le n variabili totali, m per numeri grossi

Non mi dice il perché sia una conseguenza logica.

Quindi sappiamo che esiste ma non si fa mai.

6.7 Deduzione semantica

6.7.1 Teorema di DS

Questo teorema è molto importante per dare il senso all'implicazione materiale.

$$\Gamma \Vdash F \implies G \iff \Gamma, F \Vdash G$$

▼ Dim Destra

Teorema (di deduzione semantica):
per ogni formula F e G si ha $\Gamma \Vdash F \Rightarrow G$ sse $\Gamma, F \Vdash G$.

Dimostrazione:

- Parte $\Rightarrow$:
  Per ipotesi $\Gamma \Vdash F \Rightarrow G$.
  Ovvero in ogni mondo v tale che $v \Vdash \Gamma$ si ha
  $[\![F \Rightarrow G]\!]^v = \max\{1 - [\![F]\!]^v, [\![G]\!]^v\} = 1$
  ovvero $[\![F]\!]^v = 0$ oppure $[\![G]\!]^v = 1$.
  Dobbiamo dimostrare che in ogni mondo v tale che
  $v \Vdash \Gamma, F$ si ha $[\![G]\!]^v = 1$.
  Sia v un mondo generico ma fissato.
  Per ipotesi si ha $v \Vdash F$ ovvero $[\![F]\!]^v = 1$.
  Quindi necessariamente $[\![G]\!]^v = 1$.

▼ Dim Sinistra

Teorema (di deduzione semantica):
per ogni formula F e G si ha $\Gamma \Vdash F \Rightarrow G$ sse $\Gamma, F \Vdash G$.

Dimostrazione:

- Parte $\Leftarrow$:
  Per ipotesi $\Gamma, F \Vdash G$.
  Ovvero in ogni mondo v tale che $v \Vdash \Gamma, F$ si ha $[\![G]\!]^v = 1$.
  Dobbiamo dimostrare che in ogni mondo v tale che $v \Vdash \Gamma$
  si ha $[\![F \Rightarrow G]\!]^v = \max\{1 - [\![F]\!]^v, [\![G]\!]^v\} = 1$.
  Sia v un mondo generico ma fissato.
  Per ipotesi si ha $v \Vdash \Gamma$.
  Se $[\![F]\!]^v = 1$, allora $v \Vdash \Gamma, F$ e quindi necessariamente
  $[\![G]\!]^v = 1$ e $[\![F \Rightarrow G]\!]^v = 1$.
  Altrimenti, se $[\![F]\!]^v = 0$, allora $[\![F \Rightarrow G]\!]^v = 1$.

Nota: dopo aver supposto che $v \Vdash \Gamma$, sto utilizzando l'eliminazione dell'or $F \vee \neg F$ per finire la dimostrazione (questa è una tautologia).

6.7.2 Corollario DS

Teorema (di deduzione semantica, caso n-ario):
per tutte le formule $F_1, \ldots, F_n, G$ si ha

$$F_1, \ldots, F_n \Vdash G \text{ sse } \Vdash F_1 \Rightarrow \ldots \Rightarrow F_n \Rightarrow G$$

La conclusione dovrebbe essere quasi banale grazie al teorema sopra. (Questo mi diche che sono quasi **infinite le tautologie**!) Possiamo trovare cose vere in ogni mondi.

6.7.3 Teoremini

$\Vdash F \iff \neg F$ insoddisfacibile

Γ Insoddisfacibile sse $\Gamma \Vdash \bot$ e è soddisfacibile nel caso in cui non è conseguenza logica

$\Gamma \Vdash F \iff \Gamma, \neg F$, insoddisfacibili.,Gamma finito

$\neg F \equiv F \implies \bot$

7 Deduzione naturale

La deduzione naturale è un possibile sistema deduttivo che utilizza il linguaggio naturale per questo motivo più beginner friendly.

7.1 Sintassi

7.1.1 Caratteristiche (4)

$$B, D \wedge A \vdash A \wedge (B \Rightarrow C) \Rightarrow C \qquad \dfrac{\dfrac{\dfrac{[A \wedge (B \Rightarrow C)]}{B \Rightarrow C}\,{\scriptstyle \wedge_e 2} \qquad B}{C}\,{\scriptstyle \Rightarrow_e}}{A \wedge (B \Rightarrow C) \Rightarrow C}\,{\scriptstyle \Rightarrow_i}$$

Si utilizza una BNF bidimensionale per rappresentare la ramificazione di una dimostrazione in deduzione naturale (così possiamo capire bene in quale parte del ramo viene utilizzata l'ipotesi).

1. **Radice** è la conclusione, anche indicata come top.
2. **Nodi** sono rappresentate da alcune formule
3. Le **foglie** sono formule **scaricate** (con parentesi quadre), queste sono ipotesi locali che valgono solo in quel ramo (come l'analisi per l'or).

4. Le **foglie non scaricate** rappresentano le ipotesi del problema

7.1.2 La ricorsività

Come caratteristica delle BNF io opero in modo ricorsivo su sotto-alberi più semplici.

Per ogni albero utilizzo delle regole di eliminazione o di introduzione per collegare gli alberi insieme, ecco che **utilizzo le regole di introduzione ed eliminazione per collegare le regole fra di loro** e così faccio la verifica di correttezza della dimostrazione.

7.2 Regole di inferenza

7.2.1 Sintassi delle regole di inferenza

Dobbia lettura (top-down, bottom up)!

$$\frac{F_1 \dots F_n}{F} \qquad \text{(NOME REGOLA)}$$

Dove al corrispondente del numeratore si hanno le ipotesi necessarie (che nel caso siano 0 si dicono **assioma**, in modo differente rispetto all'utilizzo finora)

1. Formula di Γ sono chiamati assioni
2. Regole senza ipotesi sono assiomi

Quindi questa definizione di assioma può avere più denotazioni, (ambigua?) no! perché è un insieme più grande che comprende entrambe le possibilità.

Devo dimostrare anche la verità di quelle ipotesi, posso allargare l'albero sopra!

$$\text{La premessa } F_i \text{ verrà indicata con } \begin{array}{c}[A]\\ \vdots \\ F_i\end{array}$$

In modo più generale

$$\frac{\frac{F_1 \dots F_n}{H_1}\ (regola-1) \quad \dots \quad \frac{G_1 \dots G_m}{H_l}\ (regola-l)}{H}\ (regola-x)$$

7.2.2 Regole di inferenza (2)

Queste regole sono state per la prima volta utilizzate in Teoria assiomatica degli insiemi per i primi esercizi.

Come faccio a concludere qualcosa sapendo qualcosa?

Cosa viene ricavata da una conoscenza?

1. Regole di introduzione di un connettivo:
   ci dicono tutti i modi in cui concludere direttamente una formula con in testa un determinato connettivo
   come concludo . . . ?
2. Regole di eliminazione di un connettivo:
   ci dicono tutti i modi in cui utilizzare direttamente un'ipotesi con in testa un determinato connettivo
   cosa ricavo da . . . ?

7.2.3 Regole Bottom up e top down

Di solito le dimostrazioni sono presentate come bottom up, perché è considerato più elegante, ma di solito si lavora sulla conclusione nel caso di troppe ipotesi (due letture per BNF)

Ogni passo di inferenza ammette sempre due letture:

1. Bottom-up (dalle premesse alla conclusione):
   date le premesse $F_1, \ldots, F_n$, posso concludere F
2. Top-down (dalla conclusione alle premesse):
   per concludere F posso ridurmi a dimostrare $F_1, \ldots, F_n$

7.2.4 Correttezza di una regola

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

Poi si potrà dimostrare che si avrà conseguenza logica per regole assemblate fra di loro. (ipotesi scaricate, devono essere rappresentate con un implica, ricorda che scaricate vuol dire che hanno ipotesi locali).

7.2.5 Invertibilità di una regola

Motivo: Ci permette di dire che le regole che stiamo dimostrando saranno poi ancora conseguenze logiche per la nostra tesi finale.

- Non scegliere regole non invertibili se posso ancora utilizzare una regola invertibile

- Valutare intuitivamente la "pericolosità" di questa regola. (come se fosse un se solo se)
- (equivalenza logica fra tante formule, mentre di solito è una formula sola);

Definizione: una regola $\frac{F_1 \dots F_n}{F}$ è invertibile quando per ogni i si ha $F \Vdash F_i$. Come per la correttezza, eventuali ipotesi scaricate (p.e. H) di F_i vanno integrate con una implicazione (es. $F \Vdash H \Rightarrow F_i$).

7.3 Dimostrazione correttezza e invertibilità di regole classiche

7.3.1 AND $\wedge$

L'AND intro- corretta e invertibile.(per invertibilità, devo espandere secondo le regole della semantica).

$$\frac{A \quad B}{A \wedge B}$$

Quindi la dimostrazione della regola di introduzione dell'and è vera, posso sempre spezzare quando mi pare.

▼ Dimostrazione

Correttezza classica: $F_1, F_2 \Vdash F_1 \wedge F_2$ in quanto, per ogni mondo v, se $[\![F_1]\!]^v = [\![F_2]\!]^v = 1$ allora $[\![F_1 \wedge F_2]\!]^v = \min\{[\![F_1]\!]^v, [\![F_2]\!]^v\} = 1$.

Invertibilità classica: $F_1 \wedge F_2 \Vdash F_i$ per $i \in \{1,2\}$ in quanto, per ogni mondo v, se $[\![F_1 \wedge F_2]\!]^v = \min\{[\![F_1]\!]^v, [\![F_2]\!]^v\} = 1$ allora $[\![F_i]\!]^v = 1$ per $i \in \{1,2\}$.

AND elim - La regola di eliminazione ci permette di utilizzare una ipotesi a scelta collegate con il connettivo dell'and

$$\frac{F_1 \wedge F_2}{F_1}\ (\wedge_{e_1}) \qquad \frac{F_1 \wedge F_2}{F_2}\ (\wedge_{e_2})$$

Regola di eliminazione classica

▼ Dimostrazione

Correttezza classica $F_1 \wedge F_2 \Vdash F_i$ per $i \in \{1,2\}$ in quanto in ogni mondo v tale che $[\![F_1 \wedge F_2]\!]^v = \min\{[\![F_1]\!]^v, [\![F_2]\!]^v\} = 1$ si ha $[\![F_i]\!]^v = 1$ per $i \in \{1,2\}$

▼ Non invertibilità

Le due regole non sono invertibili: esempio $F_1 = \top$ e $F_2 = \bot$: si ha $\top \not\Vdash \top \wedge \bot$.

$$\frac{\begin{array}{cc} & [F_1][F_2] \\ & \vdots \\ F_1 \wedge F_2 & F_3 \end{array}}{F_3}\ (\wedge_e)$$

Formula di eliminazione più generale

▼ Dimostrazione

Ricorda associatività a destra di $\Longrightarrow$

Correttezza classica: $F_1 \wedge F_2, F_1 \Rightarrow F_2 \Rightarrow F_3 \Vdash F_3$ in quanto, per ogni mondo v tale che $\min\{[\![F_1]\!]^v, [\![F_2]\!]^v\} = 1$ (e quindi $[\![F_1]\!]^v = [\![F_2]\!]^v = 1$) e $[\![F_1 \Rightarrow F_2 \Rightarrow F_3]\!]^v = 1$ (e quindi $\max\{1 - [\![F_1]\!]^v, 1 - [\![F_2]\!]^v, [\![F_3]\!]^v\} = \max\{0, 0, [\![F_3]\!]^v\} = 1$) si ha $[\![F_3]\!]^v = 1$.

▼ Non invertibilità parziale

La regola non è invertibile. Esempio: $F_3 = \top$ e $F_1, F_2 = \bot$: si ha $\top \not\Vdash \bot \wedge \bot$
La regola è invertibile se si assume $F_1 \wedge F_2$: ovvio in quanto $F_3 \Vdash F_1 \Rightarrow F_2 \Rightarrow F_3$

Ma si può vedere che non sia invertibile

7.3.2 OR $\vee$

Introduzione

▼ Enunciato

Regole di introduzione:

$$\frac{F_1}{F_1 \vee F_2} \quad (\vee_{i_1})$$

$$\frac{F_2}{F_1 \vee F_2} \quad (\vee_{i_2})$$

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)

▼ Correttezza

Correttezza: $F_i \Vdash F_1 \vee F_2$ per $i \in \{1, 2\}$ in quanto in ogni mondo v tale che $[\![F_i]\!]^v = 1$ si ha $\max\{[\![F_1]\!]^v, [\![F_2]\!]^v\} = 1$.

▼ Non invertibilità

Le due regole non sono invertibili: per esempio per $F_1 = \bot$ e $F_2 = \top$ si ha $\bot \vee \top \not\Vdash \bot$.

Perché è invertibile in un caso, e non nell'altro utilizzando le regole di eliminazione dell'OR ma non è ancora conseguenza logica)

Questa regola non è invertibile! Spesso non va bene utilizzarlo.

▼ Esempio non-invertibilità.

Consideriamo l'opzione $A \vee \neg A$, quest è conseguenza logica in tutti mondi, la dimostrazione è molto semplice. quindi $\Vdash A \vee \neg A$

Però A / quello di sopra, non è vero in tutti i mondi, quindi possiamo dire che le due non sono equivalenti, quindi non è invertibile.

Quindi $\nVdash A$ e $\nVdash \neg A$.

La best practice è utilizzare le ipotesi, e la top down non vale sempre, bisogna avere più ipotesi....

Eliminazione

In generale mi devo ridurre a ragionare nei mondi in cui solamente F1 o F2 valgono (un mondo più particolare), e poi ricompongo per dimostrare F3.

perché è possibile restringersi su un mondo particolare prima di analizzarli? Perché alla fin fine li sto analizzando tutti, ma in tempi (o rami diversi)

È una regola molto molto invertibile, quindi è da utilizzare subito!

▼ Enunciato

Regole di eliminazione:

$$\frac{F_1 \vee F_2 \qquad \begin{matrix}[F_1]\\ \vdots \\ F_3\end{matrix} \qquad \begin{matrix}[F_2]\\ \vdots \\ F_3\end{matrix}}{F_3} \ (\vee_e)$$

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

▼ Correttezza

Correttezza: si ha $F_1 \vee F_2, F_1 \Rightarrow F_3, F_2 \Rightarrow F_3 \Vdash F_3$ in quanto in ogni mondo v tale che $[\![F_1 \vee F_2]\!]^v = \max\{[\![F_1]\!]^v, [\![F_2]\!]^v\} = 1$ e
$[\![F_1 \Rightarrow F_3]\!]^v = \max\{1 - [\![F_1]\!]^v, [\![F_3]\!]^v\} = 1$ e
$[\![F_2 \Rightarrow F_3]\!]^v = \max\{1 - [\![F_2]\!]^v, [\![F_3]\!]^v\} = 1$ si ha $[\![F_3]\!]^v = 1$ (il massimo vale 1 sse c'è un $[\![F_i]\!]^v = 1$ e in tal caso
$1 = \max\{1 - [\![F_i]\!]^v, [\![F_3]\!]^v\} = \max\{0, [\![F_3]\!]^v\} = [\![F_3]\!]^v$).

▼ Invertibilità simile a AND

Invertibilità: la regola non è invertibile (controesempio: $F_3 = \top$ e $F_1 = F_2 = \bot$). Tuttavia, quando $F_1 \vee F_2$ è dimostrabile, allora la regola è banalmente invertibile.

7.3.3 Bottom e Top

Bottom

▼ Enunciato

Regole di introduzione: NESSUNA.

Regole di eliminazione:

$$\frac{\bot}{F} \quad (\bot_e)$$

Lettura bottom-up: dal falso segue qualunque cosa.

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

Si può notare che c'è l'armonia anche qua, non c'è nessun caso di introduzione e quindi non ho ipotesi nell'eliminazione.

Questa non è una regola invertibile, ossia non è vero che $F \nvdash \bot$ perché bot è sempre falso.

Top

Il Top è un assioma, unico assioma in questa sintassi in quanto non ho bisogno di ipotesi per dimostrarlo. (Notare che non è una foglia).

▼ Enunciato

Regole di introduzione:

$$\frac{}{\top}\ (\top_i)$$

Regola di eliminazione (INUTILE):

$$\frac{\top \quad F}{F}\ (\top_e)$$

Lettura bottom-up di $\top_i$: il $\top$ è vero.

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

Scrittura informale (sempre omessa) In Matita:
[$\top$ vale] by I done.

L'eliminazione del top è inutile, perché è già insita l'ipotesi in un altro, però puoi notare che è invertibile. infatti $F \Vdash \top$

7.3.4 Implicazione materiale

Introduzione

Questa regola è molto forte, sia corretta sia invertibile, perché la dimostrazione possiede sia LHS sia RHS le stesse cose quasi

▼ Enunciato

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

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

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

▼ Dimostrazione invertibilità e correttezza

$F_1 \implies F_2 \Vdash F_1 \implies F_2$ ovvia....

Eliminazione

La cosa strana è che in questo caso devo dimostrare anche l'ipotesi.

Ecco che questa non è invertibile... È la cosa che mi rende difficile la dimostrazione perché dopo quelle regole invertibili ho queste ipotesi con implicazioni e non è sempre ovvio.

▼ Enunciato

Regole di eliminazione:

$$\frac{F_1 \Rightarrow F_2 \quad F_1}{F_2} \ (\Rightarrow_e \text{ o MODUS PONENS})$$

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

Lettura top-down: per dimostrare F_2 debbo trovare un F_1 che valga e tale per cui $F_1 \Rightarrow F_2$

▼ Correttezza

Correttezza: $F_1 \Rightarrow F_2, F_1 \Vdash F_2$ in quanto in ogni mondo v tale che $[\![F_1]\!]^v = 1$ e $[\![F_1 \Rightarrow F_2]\!]^v = \max\{1 - [\![F_1]\!]^v, [\![F_2]\!]^v\} = \max\{0, [\![F_2]\!]^v\} = [\![F_2]\!]^v = 1$ si ha $[\![F_2]\!]^v = 1$.

▼ Non invertibilità

La regola non è invertibile per esempio quando $F_2 = \top$ e $F_1 = \bot$. Rimane non invertibile anche sapendo che $F_1 \Rightarrow F_2$ valga.

7.3.5 Negazione

Questa è quello che creerà la necessità di una altra logica, perché il not me lo rende complesso..

Si può dire che Non F è una altra denotazione di questa implicazione: $F \implies \bot$ perché gli unici casi in cui vale questo è che le ipotesi siano false.

Introduzione

▼ Enunciato

$F_1 \implies \bot \Vdash \neg F_1$

Regole di introduzione:

$$\frac{\begin{array}{c}[F_1] \\ \vdots \\ \bot\end{array}}{\neg F_1} \quad (\neg_i)$$

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

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

▼ Invertibilità

Molto simile a quello sopra, riesco a dimostrare l'assurdo

Eliminazione

▼ Enunciato

Regole di eliminazione:

$$\frac{\neg F_1 \quad F_1}{\bot} \quad (\neg_e)$$

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

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

Quando riesco a dimostrare l'assurdo posso dimostrare qualunque cosa, la teoria è inconsistente.

Questo mi elimina il Not, però mi rende tutto inconsistente.!! In questo ramo tutto diventa invertibile! !! Diventa solamente un gioco meccanico.

7.3.6 RAA Reductium ad abdsurdum

Questa regola è molto simile all'introduzione del not ed è necessario per avere la completezza per la deduzione semantica

▼ Enunciato

$$\dfrac{\begin{array}{c}[\neg F]\\ \vdots\\ \bot\end{array}}{F}\ (RAA)$$

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 $\bot$.

▼ Dimostrazione

$$\dfrac{\begin{array}{c}[\neg F]\\ \vdots\\ \bot\end{array}}{F}\ (RAA)$$

Correttezza classica: da $\neg F \Rightarrow \bot \Vdash F$. Infatti sia v tale che $[\![\neg F \Rightarrow \bot]\!]^v = \max\{1 - [\![\neg F]\!]^v, [\![\bot]\!]^v\} = \max\{1 - (1 - [\![F]\!]^v), 0\} = [\![F]\!]^v = 1$. Si ha $[\![F]\!]^v = 1$.

Invertibilità classica: $F \Vdash \neg F \Rightarrow \bot$ in quanto in tutti i mondi v in cui $[\![F]\!]^v = 1$ si ha $[\![\neg F \Rightarrow \bot]\!]^v = \max\{1 - [\![\neg F]\!]^v, [\![\bot]\!]^v\} = \max\{1 - (1 - [\![F]\!]^v), 0\} = 1$.

7.4 Derivabilità

Intuitivamente

Queste regole di derivabilità sono molto utili per stabilire l'eguaglianza fra regole diverse (quando una è derivabile dall'altra e viceversa..

Definizione: un insieme di regole $\mathcal{R}$ è derivabile a partire da un insieme di regole $\mathcal{S}$ quando per ogni regola in $\mathcal{R}$ le cui premesse sono $F_1, \ldots, F_n$ e la cui conclusione è F si ha $F_1, \ldots, F_n \vdash F$ usando solamente le regole in $\mathcal{S}$.

7.4.1 Dimostrazione per induzione strutturale

Intuitivamente:

Date due insieme delle regole, posso fare la dimostrazione utilizzando le regole con l'altro insieme e la dimostrazione è uguale.

Teorema: 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 $\mathcal{S}$.
Dimostrazione: per induzione strutturale sull'albero di derivazione. In tutti i casi, per ipotesi induttiva esistono alberi di derivazione per ognuna delle premesse che usano solo regole in $\mathcal{S}$. Per ipotesi esiste un albero di derivazione per la regola sotto esame che usa solo regole in $\mathcal{S}$. Componendo gli alberi si ottiene la prova voluta.

7.4.2 Derivabilità delle eliminazioni di AND

Questo teorema e dimostrazione è molto utile per stabilire l'equaglianza delle due regole, in altre parole ci sta dicendo che le due regole siano identiche.

▼ Enunciato e dimostrazione

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

Prima parte: $\{\wedge_{e_1}, \wedge_{e_2}\}$ è derivabile a partire da $\{\wedge_e\}$.

$$\frac{F_1 \wedge F_2 \quad [F_1]}{F_1}(\wedge_e) \qquad \frac{F_1 \wedge F_2 \quad [F_2]}{F_2}(\wedge_e)$$

Seconda parte: $\{\wedge_e\}$ è derivabile a partire da $\{\wedge_{e_1}, \wedge_{e_2}\}$.

$$\begin{array}{c} \frac{F_1 \wedge F_2}{F_1}(\wedge_{e_1}) \quad \frac{F_1 \wedge F_2}{F_2}(\wedge_{e_2}) \\ \vdots \\ F_3 \end{array}$$

NOTA: per la seconda parte sto prendendo in esame solamente un sotto-albero di interesse.

7.5 Armonia delle regole

Sembra che il numero delle regole di eliminazione corrisponda con il numero di regole di introduzione per ogni connettivo. (e ognuno viene corrisposto)

Eliminazione ha un caso per ogni caso di introduzione e questa utilizza le regole di introduzione.

NOTA: questo principio è molto utile per guidarci nella creazione di regole

7.5.1 Armonia OR

▼ Slide

$$\frac{F_1}{F_1 \vee F_2}\ (\vee_{i_1}) \qquad \frac{F_2}{F_1 \vee F_2}\ (\vee_{i_2}) \qquad \frac{\begin{array}{ccc}[F_1] & & [F_2] \\ \vdots & & \vdots \\ F_1 \vee F_2 & F_3 & F_3\end{array}}{F_3}\ (\vee_e)$$

- ci sono 2 modi diretti per introdurre $F_1 \vee F_2$
- nel modo *i*-esimo si ha come premessa F_i
- la regola di eliminazione analizza come la premessa $F_1 \vee F_2$ viene ricavata
- la regola ha 2 premesse: la *i*-esima assume F_i

7.5.2 Armonia AND

▼ Slide

$$\frac{F_1 \quad F_2}{F_1 \wedge F_2}\ (\wedge_i) \qquad \frac{\begin{array}{cc}[F_1][F_2] \\ \vdots \\ F_1 \wedge F_2 \quad F_3\end{array}}{F_3}\ (\wedge_e)$$

- c'è 1 modo diretto per introdurre $F_1 \wedge F_2$
- si ha come premesse F_1 e F_2
- la regola di eliminazione analizza come la premessa $F_1 \wedge F_2$ viene ricavata
- la regola ha 1 premessa e assume sia F_i che F_2

7.6 Teorema completezza e correttezza

meglio in Connettivi Logici, correttezza, variabili

7.6.1 Correttezza

▼ Enunciato

$$\text{Teorema di correttezza: } \forall \Gamma, F.\, \Gamma \vdash F \Rightarrow \Gamma \Vdash F$$

Il teorema di correttezza stabilisce la correttezza di tutte le regole date

▼ Notenotenote

Dimostrazioni di conseguenza logica

Devi fissa il mondo porre coso giutsto e poi utilizzare la semantica del mondo.

7.7 Deduzione naturale in logica di primo ordine

Possiamo estendere la deduzione naturale con alcune regole di $\forall, \exists$ qui

8 Connettivi logici

8.1 Dimostrazione teorema invarianza

8.1.1 Introduzione

Basi: Due proposizioni sono equivalenti quando valgono sugli stessi mondi.

quindi $\forall v, [\![F]\!]^v \equiv [\![G]\!]^v$.

Vogliamo dire che dati un buco presente in una proposizione, queste valgono sempre, sono in effetti equivalenti. Il buco la prendo come una variabile proposizionale. (riempire = rimpiazare il buco)

8.1.2 Operazione di sostituzione

Si può notare che ci sono 4 casi base, mentre le altre 4 sono per ricorsione strutturale.

Definizione (per ricorsione strutturale su F) di sostituzione una formula G al posto di A in F (scritto $F[G/A]$):

$$\begin{array}{ll} \bot[G/A] = \bot & (\neg F)[G/A] = \neg F[G/A] \\ \top[G/A] = \top & (F_1 \wedge F_2)[G/A] = F_1[G/A] \wedge F_2[G/A] \\ A[G/A] = G & (F_1 \vee F_2)[G/A] = F_1[G/A] \vee F_2[G/A] \\ B[G/A] = B & (F_1 \Rightarrow F_2)[G/A] = F_1[G/A] \Rightarrow F_2[G/A] \end{array}$$

8.1.3 Enunciato

Teorema di invarianza per sostituzione: per tutte le formule F, G_1, G_2 e per ogni A, se $G_1 \equiv G_2$ allora $F[G_1/A] \equiv F[G_2/A]$

La funzione di sostituzione ci permette di utilizzare una sostituzione anche nel profondo di un albero di deduzione naturale, però non è accettabile poi in sede d'esame utilizzare questo teorema per fare deduzione naturale.

La dimostrazione è per induzione strutturale abbastanza banale dopo aver definito la sostituzione, ma comunque resta un buon esercizio che dovresti fare.

8.1.4 Osservazione

8.2 Connettivi logici

8.2.1 Definizione semantica (denotazione)

▼ Enunciato

Ogni connettivo n-ario viene definito da una tabella di verità con 2^n righe.

Equivalentemente, un connettivo n-ario è una funzione $f : \{0,1\}^n \to \{0,1\}$

Dalla definizione di connettivo unario si può dedurre che esistono 2^{2^n} connettivi possibili (2^n funzioni possibili per scelte dominio e 2 scelte posssibili per codominio.

Es, per tutte le 2^n righe, devo dare in output un numero. Quindi posso dare una funzione che passa tutti 0 per tutte le righe, fino alla righa che da tutti uno per tutte le 2^n righe, finendo per avere 2^{2^n} funzioni possibili. Dobbiamo ora scegliere il perché abbiamo scelto questi connettivi fra tutti quelli presenti

8.2.2 Giustificazione delle scelte

Zero: Abbiamo preso tutti i connettivi zeroari, identificano il concetto di giusto o falso.

Uno: abbiamo preso solamente il connetivo not. (uno è uguale all'input, gli altri due la ignorano, uno la ribalta, per questo abbiamo scelto solo il not).

Binari questi sono tanti, ma non abbiamo dato una connotazione solo ad alcuni (eliminando tutti quelli banali tipo uguale a input, o inverso di input, o bot e top).

8.2.3 Riduzione fra connettivi (classico)

Questo concetto è molto simile al ruolo di equivalenza fra due regole diverse (l'eliminazione dell'and e e1 e2). in Deduzione naturale.

▼ Definizioni

Definizione: un insieme di connettivi è ridondante se contiene un connettivo riducibile ai restanti

Definizione: un insieme di connettivi è funzionalmente completo se ogni altro connettivo è riducibile a questi

Notazione: siano S e T insiemi di connettivi; scriviamo $S \triangleright T$ quando ogni connettivo di S è riducibile ai connettivi di T.

Teorema: se S è funzionalmente completo e $S \triangleright T$, allora T è funzionalmente completo.
Dimostrazione: ogni connettivo è riducibile a S poichè S è funzionalmente completo. Poichè ogni connettivo di S è esprimibile solo con connettivi di T, allora ogni connettivo è riducibile a T. Quindi T è funzionalmente completo.

Qui viene introdotto il concetto di **funzionalmente completo** che abbiamo utilizzato in Porte Logiche.

E anche il concetto di **riduzione** indicato con $\triangleright$

Studiamo quali connettivi sono necessari, scopriamo che nor e nand sono sufficienti, tutto si potrebbe ridurre a questi. Sono completi anche $\vee, \wedge, \neg, \bot, \top, \implies$ ridondante, ma funzionalmente completo

8.2.4 Motivi della scelta dei connettivi (3)

▼ Lista motivi

1. La scelta è un compromesso fra l'esigenza di considerare un insieme piccolo (ma funzionalmente completo) e un insieme che permetta di catturare naturalmente, direttamente espressioni del linguaggio naturale.
2. La nozione di riduzione fra connettivi è dipendente dalla semantica: per esempio nella semantica intuizionista (che vedremo) l'insieme $\{\vee, \Rightarrow, \neg\}$ non sarà ridondante.
3. I connettivi scelti hanno una rilevanza matematica/informatica. P.e.: $(\wedge, \vee, \bot, \top, \neg, \Rightarrow)$ vengono usati direttamente per definire $(\cap, \cup, \emptyset, \cdot, \bar{\cdot}, \subseteq)$.

Poi c'è una lunghissima lista di proprietà possibili. Il prof. in classe ha dato l'intuizione del conceto di dualità fra top e bottom e and e or (basta ordinare la retta fra -1 e 0 per verificare che corrispondono, in pratica per la definizione attuale della nostra semnatica si hanno questi valori)

8.2.5 Proprietà dei connettivi (9)

Queste scelte sono **"funzionalmente complete"** per la relazione di equivalenza.

CAIDANA

▼ Proprietà

Commutatività':
$A \vee B \equiv B \vee A, \quad A \wedge B \equiv B \wedge A$
Associatività':
$A \vee (B \vee C) \equiv (A \vee B) \vee C, \quad A \wedge (B \wedge C) \equiv (A \wedge B) \wedge C$
Idempotenza:
$A \vee A \equiv A, \quad A \wedge A \equiv A$
Distributività:
$A \vee (B \wedge C) \equiv (A \vee B) \wedge (A \vee C), \quad A \wedge (B \vee C) \equiv (A \wedge B) \vee (A \wedge C)$
Assorbimento:
$A \vee (A \wedge B) \equiv A, \quad A \wedge (A \vee B) \equiv A$
Elemento neutro:
$A \vee \bot \equiv A, \quad A \wedge \top \equiv A$
Annichilamento:
$A \vee \top \equiv \top, \quad A \wedge \bot \equiv \bot$

Doppia negazione:
$\neg\neg A \equiv A$
De Morgan:
$\neg(A \vee B) \equiv \neg A \wedge \neg B, \quad \neg(A \wedge B) \equiv \neg A \vee \neg B$

Nota: quelle in rosso non varranno per la semantica intuizionista che vedremo in seguito!

Infatti grazie a questo si può creare un **teorema di completezza** per le regole, ovvero si può dimostrare che

$P \equiv Q$ partendo solamente da queste regole, però non servono spesso per il calcolo. (a volte queste regole complicano la forma originale perché la forma di assorbimento di dovrebbe espandere).

Altre cose sono

Modus barbara e risoluzione.

8.3 Correttezza e completezza

La correttezza la saltiamo, perché è troppo complicato ed enunciata in breve sul teorema di completezza di sopra.

8.3.1 Correttezza

$\Gamma \vdash F \implies \Gamma \Vdash F$

▼ Enunciato più corretto

Localmente corrette significa che devono essere delle regole valide (che creano conseguenze logiche).

Ovvero significa che ho una ipotesi che funziona localmente ma non globalmente e lo scrivo come un implica.

Teorema di correttezza per la logica classica/intuizionista: se $\Gamma \vdash F$ (usando solo regole localmente corrette per la logica classica/intuizionista) allora $\Gamma \Vdash F$ in logica classica/intuizionista.

Intuizione

Ogni regola metto in un foglietto diverso (anche l'ipotesi scaricata è presente di un foglietto sopra).

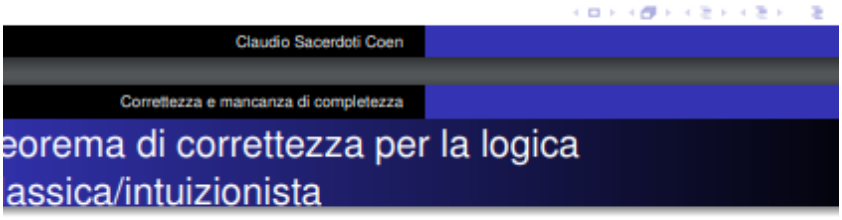
Voglio dire che quella regola non è una regola scaricata per un foglietto sotto. (viene scaricata solamente da un foglietto sopra.

▼ Dimostrazione

Dimostrazione: per induzione strutturale sull'albero di derivazione $\Gamma \vdash F$.

Caso A: poichè A è una foglia non cancellata, si ha $A \in \Gamma$. Pertanto $\Gamma \Vdash A$.

Caso $[A]$: impossibile in quanto un'ipotesi viene scaricata solamente da una regola.



Caso $\frac{T_1 \dots T_n}{F}(r)$ dove $T_1, \dots, T_n$ sono i sottoalberi immediati dell'albero di deduzione:
Sia T_i la derivazione $\Theta_i \vdash F_i$. Si ha $\Theta_i = \Gamma_i \cup \Delta_i$ dove Δ_i è l'insieme delle ipotesi cancellate in T_i dalla regola r, Γ_i è l'insieme delle ipotesi non cancellate dalla regola r e $\Gamma \supseteq \bigcup_i \Gamma_i$.
Per ipotesi induttiva, $\Theta_i = \Gamma_i \cup \Delta_i \Vdash F_i$ per ogni i. Per correttezza locale della regola r si ha
$\Delta_1 \Rightarrow F_1, \dots, \Delta_n \Rightarrow F_n \Vdash F$. Per il teorema di deduzione semantica, da $\Delta_i \cup \Gamma_i \Vdash F_i$ consegue che $\Gamma_i \Vdash \Delta_i \Rightarrow F_i$. Quindi per la transitività della conseguenza semantica, si ottiene $\Gamma \supseteq \bigcup_i \Gamma_i \Vdash F$.

QED.

I **Passi principali di dimostrazione**

1. Caso base A e caso impossibile [A]
2. Ipotesi induttiva: i sotto-alberi sono conseguenza logica di ipotesi globali e ipotesi locali (localmente corrette).
3. Per utilizzo di regole locali so che F è conseguenza logica di molti implica.
4. Per il teorema di deduzione semantica trasformo l'ipotesi induttiva con implica in RHS e LHS solo ipotesi globali del ramo principale.
5. Unisco con transitività della conseguenza logica.

8.3.2 Perché correttezza più semplice di completezza

Affinché abbia la correttezza, mi bastano delle regole che siano localmente corrette, se invece ho una regola incorretta riesco a derivare bottom da top, e quindi mi creerebbe una teoria inconsistente (tutto che valga).

Vorrei dire che **ho tutte le regole per catturare un concetto semantico** utilizzando un concetto sintattico (finito). Come faccio a usare una quantità finita per catturare l'infinito? Quindi per le logiche semplici come la logica proposizionale classica si può, per le altre no.

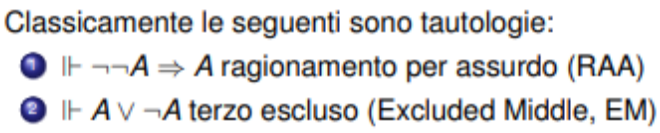
È sorprendente che un insieme finito di regole sia sufficiente a dimostrare infinite ipotesi, questo in matematica è catturato il concetto di **compattezza**.

Due ingredienti

1. Devo avere tutte le regole, queste sono sufficienti per dimostrare tutte le conseguenze logiche.
2. Da un insieme finito di regole devo essere in grado di dimostrare tutto.

8.3.3 Completezza in logica classica

▼ Non è possibile dimostrare queste classiche tautologie RAA, EM



Nel caso due, posso dimostrare A oppure non A per introduzione dell'OR. Però poi non ho niente per continuare, in certi mondi non funziona A, perché non ho ipotesi, e non ho niente per dimostrare bottom con l'introduzione della negazione. Quindi queste non sono dimostrabili.

Ma questi valori sono validi se i valori di verità sono solamente due! **le regole che ho non colgono la dualità (vero falso) della logica classica**. infatti queste si dimostrazione solamente con l'introduzione della regola RAA.

Ma l'introduzione di questa regola **toglie l'algoritmicità della dimostrazione** quindi non è da fare.

Per avere dimostrazione della correttezza della regola qui

8.4 Variabili

▼ Definizione funzione Var

Definizione: la funzione $Var(F)$ (variabili di F, variabili che occorrono in F) è definita per ricorsione strutturale su F come segue:

$$
\begin{aligned}
Var(\bot) &= \emptyset \\
Var(\top) &= \emptyset \\
Var(A) &= \{A\} \\
Var(\neg F) &= Var(F) \\
Var(F_1 \wedge F_2) &= Var(F_1) \cup Var(F_2) \\
Var(F_1 \vee F_2) &= Var(F_1) \cup Var(F_2) \\
Var(F_1 \Rightarrow F_2) &= Var(F_1) \cup Var(F_2)
\end{aligned}
$$

Costruiamo una funzione Variabile definita per ricorsione strutturale che ritorni tutte le variabili esistenti in una formula logica.

Cerchiamo di creare una sintassi che possa essere utilizzabile **tutta l'infinità dei mondi.**

Riesco quindi a introdurre il concetto di equivalenza di mondi in funzione di una proposizione.

8.4.1 Teorema: var(F) finito per ogni F

La dimostrazione (intuizione) per induzione strutturale è abbastanza easy. Nel caso dei casi finiti dovrei dimostrare che il singoletto e il vuoto sono finiti. Nel caso di parti composte, devo supporre che siano finite per le espansioni, allora l'unione di insiemi finiti è ovvia

In realtà la dimostrazione vera devi formalizzare prima il concetto di finito e non finito, che è in teoria degli insiemi, quindi hai bisogno di molte altre cose.

8.4.2 F in v usa restrizione di v al dominio Var(F)

Posso creare una relazione di equivalenza se per ogni X in Var(F) ho che v(X) = v2(X) e ho che i due mondi sono uguali.

Chiamo solamente delle variabili di un mondo?

▼ Dimo

Teorema: $[\![F]\!]^v$ usa solamente la restrizione di v al dominio $Var(F)$

Corollario: se v_1 e v_2 sono tali che per ogni $X \in Var(F)$ si ha $v_1(X) = v_2(X)$, allora $[\![F]\!]^{v_1} = [\![F]\!]^{v_2}$.

Dimostrazione del teorema: per induzione strutturale su F.

- Casi $\bot$, $\top$: non usano v.
- Caso A: $[\![A]\!]^v = v(A)$ e $Var(A) = \{A\}$.
- Caso $\neg F$:
  - Per ipotesi induttiva $[\![F]\!]^v$ usa solo la restrizione a $Var(F)$.
  - Si ha $[\![\neg F]\!]^v = 1 - [\![F]\!]^v$ e $Var(\neg F) = Var(F)$.
- Caso $F_1 \wedge F_2$ (i casi $F_1 \vee F_2$ e $F_1 \Rightarrow F_2$ sono simili):
  - Per ipotesi induttiva $[\![F_1]\!]^v$ usa solo la restrizione a $Var(F_1)$ e $[\![F_2]\!]^v$ usa solo la restrizione a $Var(F_2)$
  - Si ha $[\![F_1 \wedge F_2]\!]^v = \min\{[\![F_1]\!]^v, [\![F_2]\!]^v\}$ che usa v sulla restrizione al dominio di F_1 e a quello di F_2 e $Var(F_1 \wedge F_2) = Var(F_1) \cup Var(F_2)$.

Dimostrazione

In pratica sto collassando in una classe di equivalenza con il

Il fatto che siano finiti mi permette di costruire tabelle di verità.

Nota

Grazie a questo teorema posso dire che una variabile (proposizione) sia valida in tutti i mondi dipendentemente solamente dal valore di verità di una variabile al caso base.

Questo significa che le proposizioni logiche sono valide per tutti i mondi che soddisfino le precodizioni e non solamente nel mondo specifico!

9 Semantica intuizionista

▼ Tutte le slides

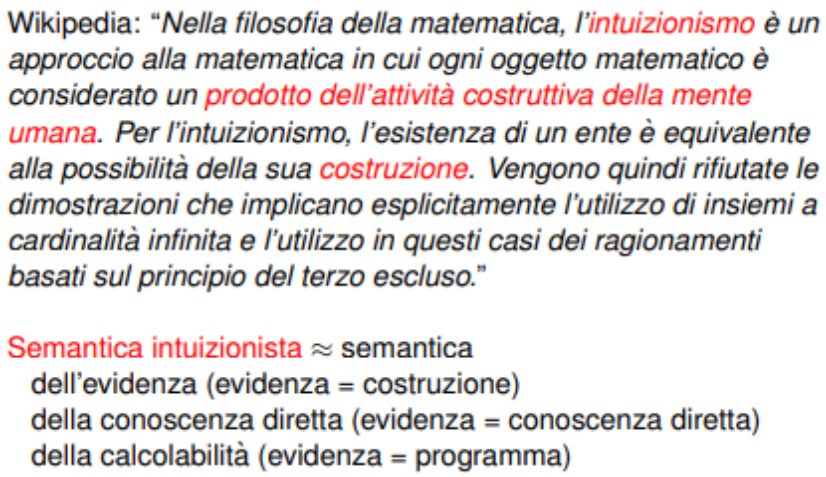
https://s3-us-west-2.amazonaws.com/secure.notion-static.com/45c78b4e-4084-43d2-89d1-597ee49349f3/slides10_202021.pdf

Molto importante questo documento per avere chiara la differenza fra la logica intuizionista e la Logica Proposizionale classica.

Questa logica intuizionista non si preoccupa del noumeno platonico, ma solo di una prova reale.

Introduzione:

▼ wikipedia



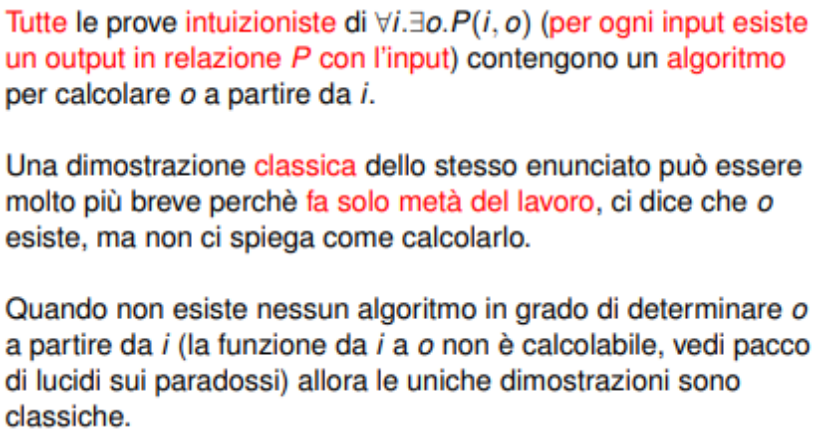
9 11 Scopi di intuizionista (3)

1. Semantica dell'evidenza → costruzione della prova
2. Semantica della conoscenza diretta = conoscenza diretta
3. Semantica della calcolabilità = programma, algoritmo della soluzione

9.1 Invenzione o scoperta

La semantica intuizionista vede la matematica come una creazione (e questa cosa interessa molto all'informatico perché è una prova., mentre la semantica classica vede la matematica come una scoperta

▼ Caratteristica principale della logica intuizionista



Esempio il paradosso di Banach-tarksi non è calcolabile (nessuno ha duplicato una sfera doro diciamo :))

Seziono la sfera in 3 parti e faccio movimenti rigidi (non deformo, posso rotare, trasportare) e dimostro che da questi movimenti rigidi ottengo due sfere con lo stesso volume e con gli stessi punti.

9.1.1 Evidenza indiretta e diretta

Nelle due logiche il significato di esistenza è differente.

1. Classica: l'esistenza, ma non so esattamente che numero sia
2. Intuizionista: l'elemento che soddisfa, riesco proprio a trovare l'elemento tale che mi soddisfi le mie proprietà.

NOTA: il fatto che esista una prova intuizionista mi permette di avere un algoritmo per tirare fuori un elemento che me lo soddisfi, mentre dalla prova classica no.

9.1.2 Effetti dimostrazione per assurdo

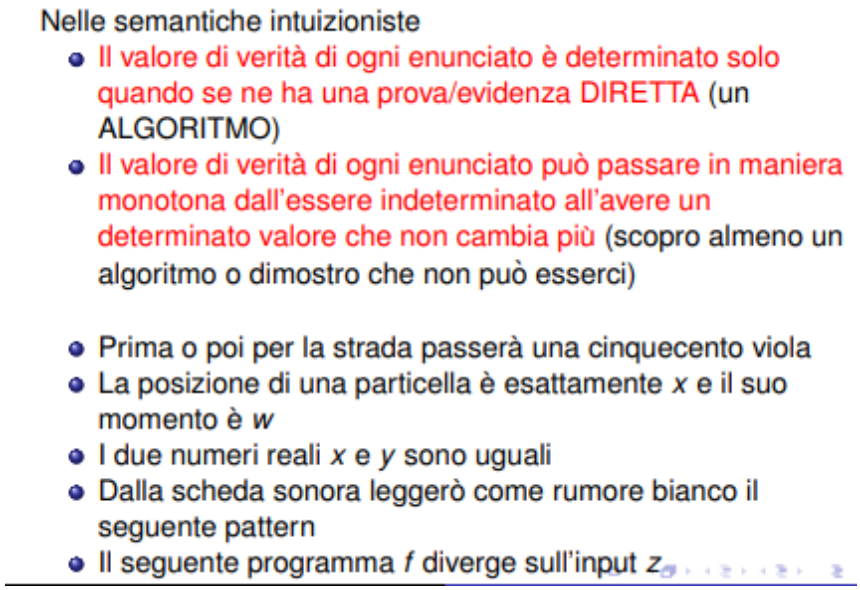
Questo metodo di dimostrazione non è stato ben accettato nell'epoca in cui è stato creato perché non permetteva il buon calcolo:

1. No calcolo
2. Molto utile per dimostrare teoremi che non si potevano dimostrare in modo intuizionistico
3. Molto veloce perché rende le dimensioni delle prove minori (ma perché fa meno lavoro!)

9.2 Enunciati e semantiche della logica intuizionista

1. **Algoritmo** l'evidenza diretta è necessaria per la determinazione del valore di verità
2. Il valore di verità è potenzialmente determinabile (ossia può diventare fissata dopo un pò di tempo), ci deve essere una algoritmo o che non esista.

▼ Enunciato ed esempi



Analizzando la prima proposizione, per la logica classica dovrei avere un valore di verità fissato, invece sono in un mondo indeterminato.

Per i numeri, l'algoritmo non riuscirebbe mai a finire a comparare due numeri periodici e non riuscirebbe a dare un risultato in tempo finito.

Mentre l'ultimo esempio è stato dimostrato in Logica meta-linguistica

Da questi si posso ricavare due semantiche:

9.2.1 Semantica di Kripke

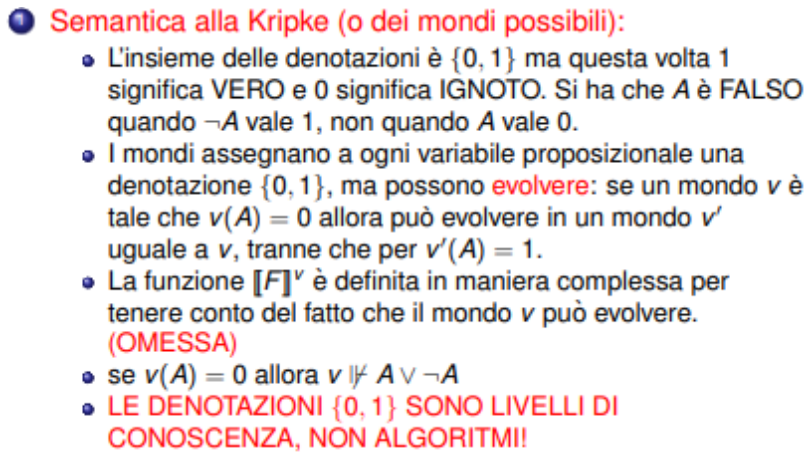
La caratteristica principale di questa semantica è che le proposizioni possono avere un range da [0,1].

E che queste denotazioni possono evolvere da 0 a 1 col tempo. Bisogna quindi avere una funzione semantica che possa trattenere il tempo.

0 è ignoto 1 è vero e questi valori evolvono.

Questi valori sono dei valori di conoscenza ma non verità, e non algoritmico!

▼ Enunciato



9.2.2 Altro

Si tratterà anche della semantica di Brouwer-Heyting-Kolmogorov subito dopo

9.3 Semantica di Brouwer-Heyting-Kolmogorov

9.3.1 Introduzione

> Data una formula F, la sua denotazione è l'insieme delle prove esplicite (algoritmi) che risolvono quella formula.

Questa semantica è molto utile per l'informatico perché le formule sono descrizioni di problemi e l'altro soluzione.

osservazioni:

Questa semantica è molto utile per comporre delle soluzioni (grazie ai connettivi). L'insieme vuoto è l'inesistenza di soluzioni algoritmiche (che potrebbe essere non più vuoto in futuro).

Qui ha senso introdurre il concetto di **dedicibilità** ovvero la possibilità di costruire un algoritmo che me lo risolva.

9.3.2 Enunciato e definizione semantica

▼ Enunciato

2. Semantica di Brouwer-Heyting-Kolmogorov:
   - Formula F = descrizione di un problema
     es. $\forall i.\exists o.P(i,o)$
   - Denotazione di F = insieme di evidenze, ovvero 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 descrive un problema a partire da altri problemi
   - La denotazione di un connettivo è un insieme di algoritmi che risolvono il problema composto usando gli algoritmi per i problemi semplici

▼ Definizione semantica

- $[\![A]\!]^v = v(A)$
- $[\![\bot]\!]^v = \emptyset$
- $[\![\top]\!]^v = \{\star\}$ ($\top$ è un problema banale e $\star$ lo risolve)
- $[\![F \wedge G]\!]^v = [\![F]\!]^v \times [\![G]\!]^v$
  (l'$\wedge$ chiede di risolvere entrambi i problemi;
  algoritmo = coppia di un algoritmo per F e uno per G)
- $[\![F \vee G]\!]^v = [\![F]\!]^v \oplus [\![G]\!]^v$
  (l'$\vee$ chiede di risolvere uno dei due problemi dicendo quale;
  algoritmo = coppia $\langle b, \pi \rangle$ dove
  $b = 0$ e π è un algoritmo per F oppure
  $b = 1$ e π è un algoritmo per G)
- $[\![F \Rightarrow G]\!]^v = [\![G]\!]^{v[\![F]\!]^v}$
  (procedure che mappano soluzioni a F in soluzioni a G)

9.3.3 Nota sul VOID:

con void in c starei restituendo la stellina (valore vero), ovvero sto restituendo sempre una sequenza giusta (eliminazione del top è inutile quindi non ci faccio caso).

9.3.4 EM e RAA in semantica intuizionista

Queste dimostrazioni valide in logica classica non hanno più senso in questo caso

▼ Esempio

1. $\not\Vdash A \vee \neg A$
   per un problema A ignoto non conosciamo nessun algoritmo che lo risolva e nemmeno possiamo dire che tale algoritmo non esista
2. Per A particolari (es. $A = n$ è pari) si può dimostrare $A \vee \neg A$ dando un algoritmo (nell'esempio: l'algoritmo ci dice se n è pari oppure no). In tal caso A si dice decidibile.
3. $\not\Vdash \neg\neg A \Rightarrow A$
   il fatto che non possa non esserci un algoritmo per un problema A ignoto non ci dà un algoritmo per A

9.3.5 Correttezza e completezza

Potrebbe essere un buon esempio confrontare la correttezza e completezza intuizionistica con quella classica.

▼ Slide

Teorema (correttezza): per ogni Γ e F, se $\Gamma \vdash F$ senza usare la RAA allora $\Gamma \Vdash F$ in logica proposizionale intuizionista.
Dimostrazione: omessa.

Ovvero: se riesco a dimostrare F a partire da Γ allora ricavo (almeno un) algoritmo che risolva il problema F usando algoritmi per i problemi in Γ (es. una libreria di codice per Γ).

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.
Dimostrazione: omessa.

Completezza forte e debole

▼ Slide

Teorema di completezza (forte) per la logica proposizionale classica: per ogni Γ, F, se $\Gamma \Vdash F$ allora $\Gamma \vdash F$.

Teorema di completezza debole per la logica proposizionale classica: per ogni Γ, F t.c. Γ sia un insieme finito, se $\Gamma \Vdash F$ allora $\Gamma \vdash F$.

La versione debole è dimostrabile usando la meta-logica intuizionista, ovvero c'è un algoritmo che dati Γ (finito) e F permette di costruire un albero di deduzione naturale per $\Gamma \vdash F$.

La versione forte è dimostrabile solo usando la meta-logica classica, ovvero l'albero di deduzione naturale per $\Gamma \vdash F$ esiste, ma non c'è un algoritmo che permette di costruirlo.

▼ Abbozzo di ragionamento per queste completezza

Intuizioni:

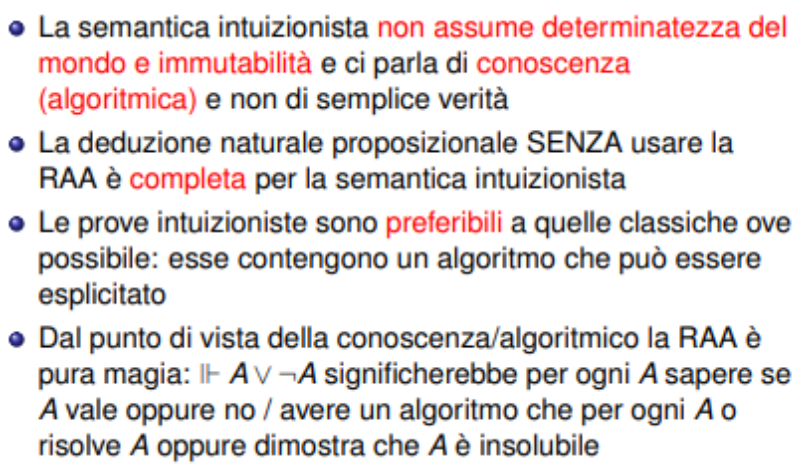
- che nel caso forte non vi sia un algoritmo è abbastanza intuitivo: l'algoritmo in un tempo finito dovrebbe saper analizzare un'infinità di ipotesi (Γ) per scegliere quali usare
- l'algoritmo per il caso debole funziona in questo modo: siano $V_1, V_2, \ldots, V_n$ le variabili che occorrono in Γ (finito!) e F. L'albero inizia per casi su $V_1 \vee \neg V_1$ (per EM), in ogni ramo va per casi su $V_2 \vee \neg V_2$, etc. fino ad andare per casi su $V_n \vee \neg V_n$. A questo punto ogni ramo ci parla di un solo mondo e si può dimostrare la tesi a partire dalle ipotesi con una prova intuizionista meccanica.
- che ci sia una dimostrazione in meta-logica classica del caso forte è sorprendente! (vedi slide dopo)

Un algoritmo in un tempo finito non può analizzare un input infinito, quindi non potrebbe dimostrarlo un algoritmo. Esiste però una dimostrazione classica.

9.3.6 Conclusioni

- Questa semantica si può evolvere nel tempo (trovando nuovi algoritmi che mi risolvano il problema)
- Si ha una completezza debole per logiche senza RAA
- Se ho una prova intuizionista riesco a costruire un algoritmo che mi risolvi un problema
- EM non ha molto senso qua, se fosse una tautologia sarebbe un risultato molto forte

▼ Slide



9.3.7 La negazione

Le formule negate non hanno informazione al loro interno.

Partiamo dalla regola del not, ovvero per dimostrare nonF devo dimostrare che F implica l'assurdo.

Ovvero devo dire che ci sia una funzione che parta da F e che arrivi a nulla.

Le soluzioni possibili sono solamente vuoto e singoletto di vuoto (in altri termini possiamo dire che abbiamo 0 e 1). chiaramente queste funzioni non sono affatto utili per cui non esiste un algoritmo che mi da cose utili.

Negare due volte è equivalente a distruggere una informazione (devi fare una tabellina con la regola per vedere che c'è questo)

notF	Stato di F	notnotF
singoletto vuoto	Uguale a Vuoto	vuoto
vuoto	Diverso da vuoto	singoletto vuoto

perché nonF mi può dare solamente due output, mi ha distrutto tutto l'algoritmo iniziale!

Dimostrare che non esiste significa dire che non esiste informazione.

Prova a dimostrare queste:

$$\neg(F_1 \vee F_2) \Vdash \neg F_1 \wedge \neg F_2$$

Mentre per quello sopra invertito (invertendo or e and di sopra) non è intuizionista, perché l'output ha più informazioni per qualche motivo strano.

9.4 Teorema di compattezza

Questo teorema è una conseguenza molto utile della proprietà di completezza, come in slide:

▼ Enunciato e dimostrazione del teorema

Teorema di compattezza: per ogni Γ, F, se $\Gamma \Vdash F$ allora esiste un $\Delta \subseteq \Gamma$, Δ finito t.c. $\Delta \Vdash F$.

Dimostrazione: siano Γ, F t.c. $\Gamma \Vdash F$. Per il teorema di completezza forte si ha $\Gamma \vdash F$. Quindi c'è un albero di deduzione naturale, necessariamene finito, che dimostra F e le cui foglie non scaricate formano un sottoinsieme finito Δ di Γ t.c. $\Delta \vdash F$. Per il teorema di correttezza si ha $\Delta \Vdash F$. Qed.

Questo teorema è molto forte, perché se ho un insieme infinito di filtri, vuol dire che ho bisogno solamente di limiti finiti. E il fatto che l'infinito si possa ridurre al finito è un fatto sorprendente.

9.4.1 Conseguenze della compattezza (4)

La cosa che era sorpredente della compattezza in questi casi è la possibilità di ritrovare in un caso infinito un caso finito da cui è possibile poter ricavare la cosa voluta!.

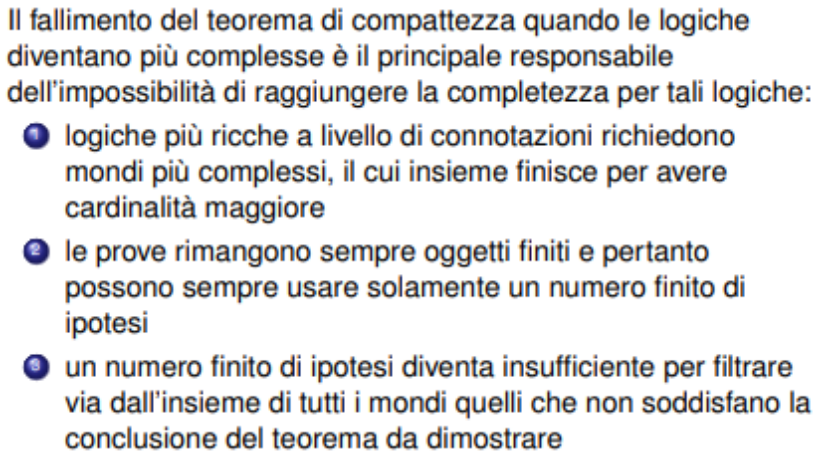
▼ Conseguenze

1. Esistono infiniti mondi
2. In $\Gamma \Vdash F$ le ipotesi Γ filtrano via i mondi fino a quando quelli che restano non soddisfano tutti F
3. Infinite ipotesi (quando Γ è infinito) permettono di filtrare via un'infinità di mondi in modo molto preciso
4. Il teorema di compattezza ci dice che in verità un numero finito di ipotesi contenute in Γ sono già sufficienti a filtrare via abbastanza mondi da rendere comunque F vera nei rimanenti

Ovviamente il teorema di compattezza è dimostrabile solo in una meta-logica classica (un algoritmo dovrebbe scegliere da infinite ipotesi Γ quelle da tenere).

9.4.2 Fallimento della compattezza per logiche complesse (3)

▼ Slide



10 Logica del primo ordine

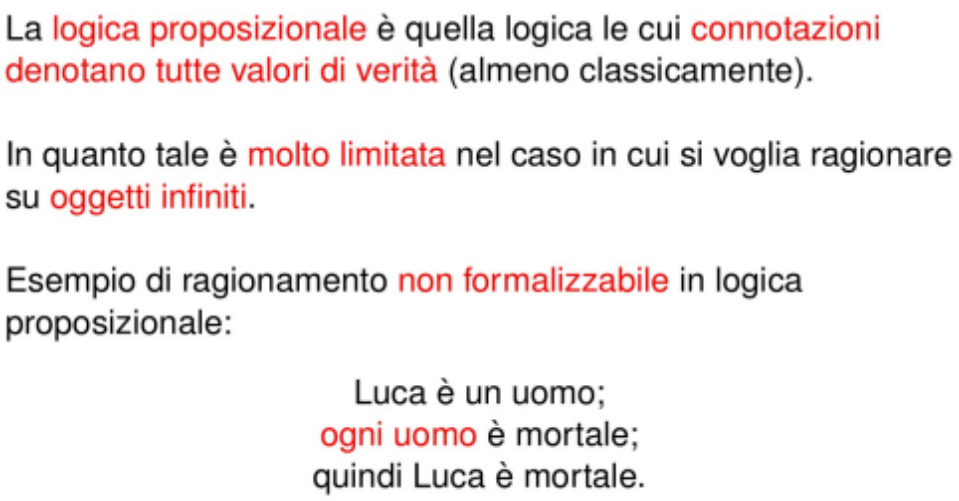
10.1 Introduzione

Questa è la logica più utilizzata dai matematici

10.1.1 Limitatezza della logica proposizionale

La logica proposizionale classica non è in grado di ragionare sull'infinito

▼ Slide



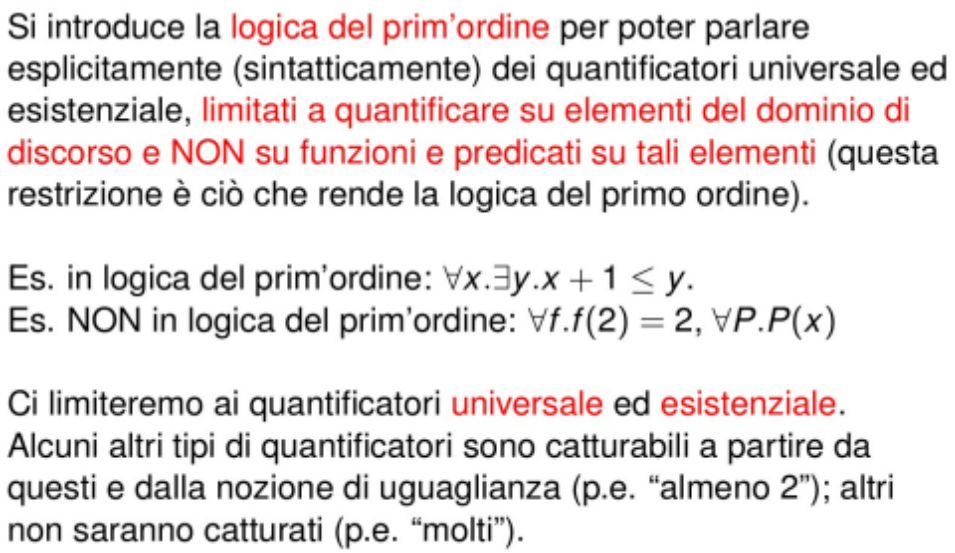
Fino ad ora abbiamo utilizzato una metalogica per giustificare il per ogni e l'esiste nelle dimostrazioni fin'ora.

Dobbiamo quindi dare una definizione più formale dei **quantificatori**.

10.1.2 Obiettivo della logica del primo ordine

Si può quindi identificare come l'obiettivo della logica di primo ordine l'introduzione dei quantificatori dell'universale e dell'esiste

▼ Slide

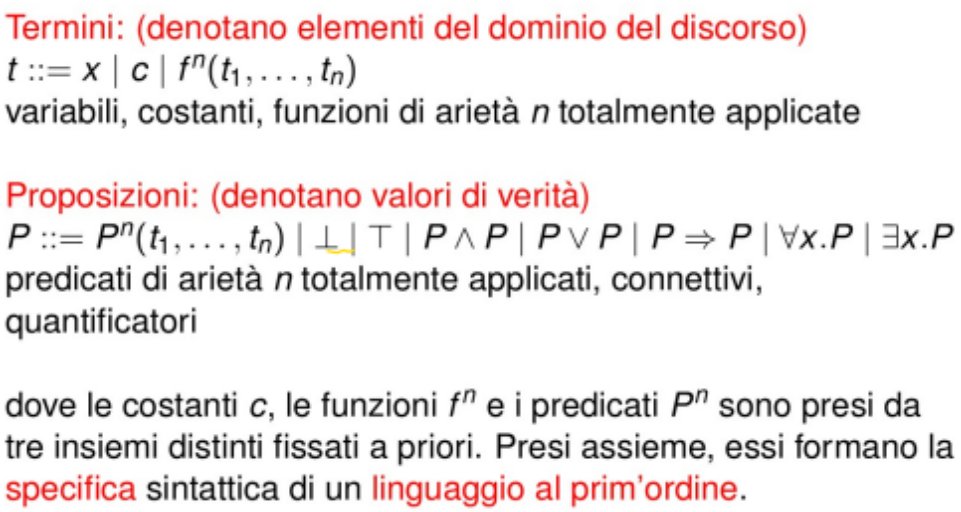


Possono denotare

10.2 Sintassi

In questa sintassi stiamo dividendo in Termini e proposizioni (le proposizioni che si possono trovare nella logica proposizionale classica).

▼ Sintassi



Come si può osservare nella sintassi di logica del primo ordine estende la logica proposizionale classica perché per P 0 ho le singole proposizioni

Quindi si divide in un **dominio di discorso**, ossia l'insieme dei termini possibili come costanti e le**formule o proposizioni** che possiedono un valore di verità.

10.2.1 Perché primo ordine

Si chiama logica di primo ordine perché non si possono utilizzare le funzioni sulle variabili nel dominio di discorso.

Questa è l'ultima logica in cui vale ancora la completezza, e la correttezza, nelle logiche superiori non sarà più possibile catturare tramite un concetto sintattico il valore semantico.

Questa è la logica di primo ordine che basta ai matematici per fare tutto (questo perché le funzioni dei matematici in realtà sono degli insiemi, non qualcosa che calcola).

10.2.2 Possibili denotazioni

▼ Possibili denotazioni

Per farlo, oltre ai quantificatori, ci servono:

- variabili $x, y, z, \ldots$
  denotano un oggetto ignoto e variabile sul dominio
- costanti $0, 1, \ldots, \pi, e, \ldots,$ marco, luca, $\ldots c, d, \ldots$
  denotano un oggetto noto e fissato del dominio
- simboli di funzione $+, *, \ldots,$ il fratello di, il figlio di $\cdot$ e $\cdot, \ldots$
  applicati a connotazioni di oggetti del dominio, denotano un oggetto del dominio
- simboli di predicato
  $<, \leq, =, \ldots,$ essere sposato con, essere vedovo, $\ldots$
  applicati a connotazioni di oggetti del dominio, denotano un valore di verità

1. Oggetti ignoti nel dominio
2. Oggetti fissati
3. Connotazioni di denotazioni oggetti
4. Connotazioni di denotazioni di valori di verità

10.3 Semantica

▼ Introduzione

La semantica di una costante dipende dal mondo.

La semantica di una funzione dipende dal mondo.

La semantica di un predicato dipende dal mondo.

La semantica di una variabile è determinata dal quantificatore che la lega.

La semantica dei connettivi è fissata.

La semantica dei quantificatori è fissata: $\forall x.P$ significa "per tutti gli x vale P" e $\exists x.P$ significa "per un qualche x vale P" senza alcun'altra restrizione.

Questa parte è approfondita dopo con mondo ed interpretazione

10.4 Binder

I binder sono un concetto fondamentale nell'informatica (soprattutto a chi andrà a fare i compilatori). Quindi stiamo astraendo un livello di semantica! Connettivi ancora più astratti.

> I binder legano una **variabile** e uno **scope** (Cattura una variabile (o serie di varaibile) in uno scope che viene valutato più e più volte).

1. Formula matematica (uno scope nel senso di derivata, integrale sommatoria e simili)
2. I Simboli logici per ogni esiste.

Vado a valutare una unica formula all'interno di uno scope (e continuo ripetutamente a sostituire e calcolare su tanti valori, e restituisco il risultato sintetizzato).

10.4.1 Shadowing

Nei binder non si può accedere alle variabili esterne se hanno lo stesso nome, si dice **shadowing** (quello interno nasconde l'esterno, quindi fa ombra, nasconde).

Un esempio è ridichiarare un parametro formale in una funzione.

10.4.2 Diagrammi di legame

Sono molto utili per capire le variabili del binder e lo scope di queste variabili

Cose da fare:

1. Legare variabile a ogni binder e lo scope
2. Esistono le variabili che non vengono mai legate, si dicono variabili libere queste (come libreria o variabili globali), queste si indicano conuna freccia all'infinito con il nome

10.4.3 Variabili libere

Queste variabili non sono modificabili (come provare a cambiare il nome di una funzione di libreria esterna).

▼ Definizione per induzione strutturale

Definizione di variabili libere (free variables): per induzione strutturale sulle proposizioni e sui termini.

$$\mathrm{FV}(x) := \{x\}$$
$$\mathrm{FV}(f^n(t_1, \ldots, t_n)) := \mathrm{FV}(t_1) \cup \ldots \cup \mathrm{FV}(t_n)$$

$$\mathrm{FV}(P^n(t_1, \ldots, t_n)) := \mathrm{FV}(t_1) \cup \ldots \cup \mathrm{FV}(t_n)$$
$$\mathrm{FV}(\bot) = \mathrm{FV}(\top) := \emptyset$$
$$\mathrm{FV}(\neg P) := \mathrm{FV}(P)$$
$$\mathrm{FV}(P \wedge Q) = \mathrm{FV}(P \vee Q) = \mathrm{FV}(P \Rightarrow Q) := \mathrm{FV}(P) \cup \mathrm{FV}(Q)$$
$$\mathrm{FV}(\forall x.P) = \mathrm{FV}(\exists x.P) := \mathrm{FV}(P) \setminus \{x\}$$

Un quantificatore, legando una variabile, la rende non più libera.

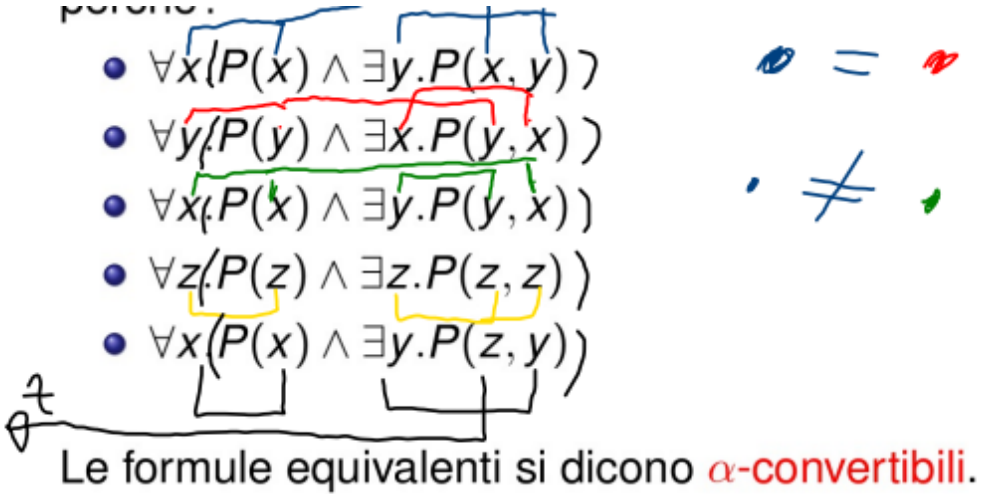
10.5 alfa-convertibilità

Si chiama alfa perché una branca dell'informatica che studiava i lambda, cuore del linguaggio di programmazione funzionale.

> Si dice che una serie di formule sono alfa-convertibili se il diagramma di legame creato dalle formule è uguale → **relazione di equivalenza**

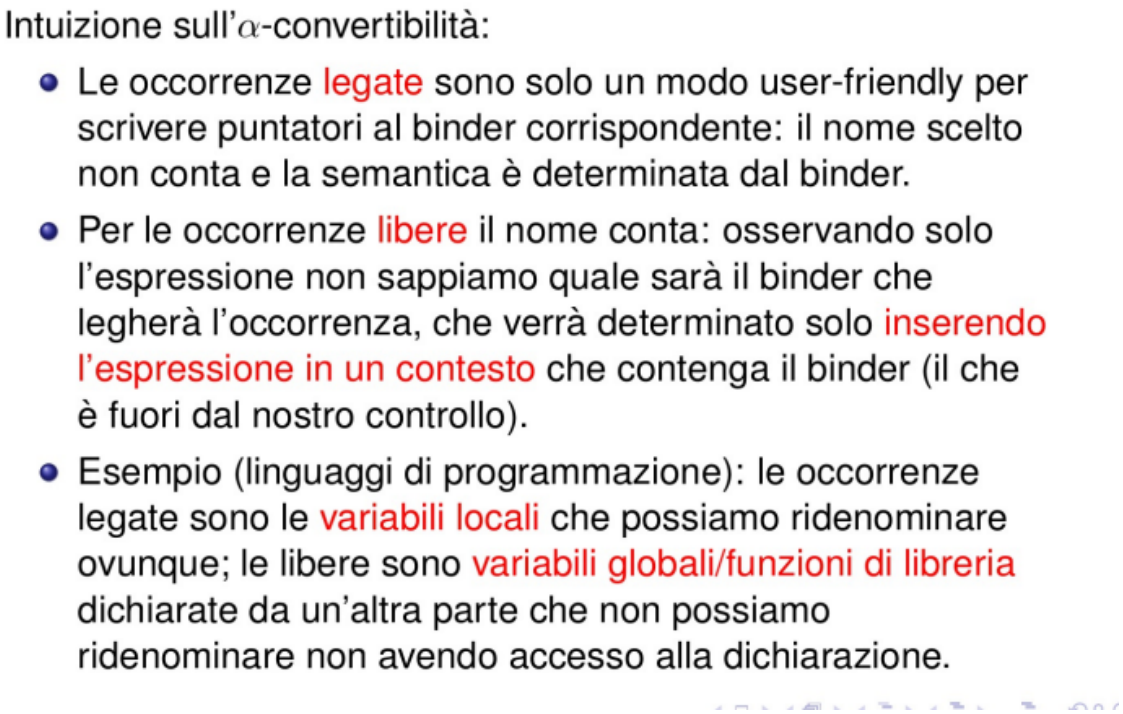
Essendo una relazione di equivalenza possiamo lavorare su una classe di equivalenza, quindi sarebbe bene ragionare al livello di **insieme quoziente delle formule**.

▼ Esempi



Nell'esempio in giallo, la Z fa shadowing, ha cambiato una variabile che in primo momento apparteneva a uno scope esterno.

▼ Altro esempio



10.5.1 Sostituzione in logica primo ordine

Praticamente questa nozione ci dice che una funzione ha lo stesso output anche se quello che ci va dentro è una variabile con un nome diverso. Questa nozione ha una certa similitudine con la funzione di sostituzione, in quanto entrambi parlano di invarianza sulla sostituzione di variabili.

La differenza principale è che questa parla di binder mentre quella di prima parla di stesso valore di verità di una proposizione.

Possiamo allora definire una funzione di sostituzione anche per la logica del primo ordine che faccia attenzione anche ai diagrammi di flusso e i binder

▼ Sostituzione

Definizione di sostituzione di un termine per una variabile: per induzione strutturale sul termine in cui avviene la sostituzione.

$$x[t/x] = t$$
$$y[t/x] = y$$
$$f^n(t_1, \ldots, t_n)[t/x] = f^n(t_1[t/x], \ldots, f_n[t/x])$$

$f[t/x]$

$$P^n(t_1, \ldots, t_n)[t/x] = P^n(t_1[t/x], \ldots, f_n[t/x])$$
$$\bot[t/x] = \bot$$
$$\top[t/x] = \top$$
$$(\neg F)[t/x] = \neg(F[t/x])$$
$$(F_1 \wedge F_2)[t/x] = F_1[t/x] \wedge F_2[t/x]$$
$$(F_1 \vee F_2)[t/x] = F_1[t/x] \vee F_2[t/x]$$
$$(F_1 \Rightarrow F_2)[t/x] = F_1[t/x] \Rightarrow F_2[t/x]$$
$$(\forall x.P)[t/x] = \forall x.P$$
$$(\forall y.P)[t/x] = \forall z.(P[z/y][t/x]) \text{ per } z \notin FV(t) \cup FV(P)$$
$$(\exists x.P)[t/x] = \exists x.P$$
$$(\exists y.P)[t/x] = \exists z.(P[z/y][t/x]) \text{ per } z \notin FV(t) \cup FV(P)$$

10.6 Mondo o interpretazione

Non è più sufficiente avere un mondo indicato solamente come una v, ma **è necessaria una coppia ordinata**: (A, I) dove A è l'insieme non vuoto di denotazioni, mentre I è simile alla vecchia funzione semantica v, però associa degli elementi in A altri elementi in A, non dice niente sulle connotazioni.

▼ Esempio

Definizione: un mondo o interpretazione per la logica del prim'ordine è una coppia (A, I) dove A è un insieme non vuoto di denotazioni per i termini e I è una funzione di interpretazione che associa

- a ogni funzione f^n una funzione il cui dominio è $A^n = A \times \ldots \times A$ (n volte) e il cui codominio è A
- Caso particolare: per ogni costante c, $I(c) \in A$
- a ogni predicato P^n una funzione il cui dominio è A^n e il cui codominio è $\{0, 1\}$ o, equivalentemente, un sottoinsieme di A^n
- Caso particolare: per ogni predicato 0-ario P, $I(P) \in \{0, 1\}$ come nel caso della logica proposizionale

Nota: un mondo non è più rappresentabile come una sequenza di booleani e non è più possibile usare tabelle di verità.

10.6.1 Nozione semantica di per ogni ed esiste

▼ Esempi di formalizzazione sintattica errata

Come catturare la nozione di variazione di x sul dominio A?
Vediamo prima un paio di modi non corretti:

- $[\![\forall x.P]\!]^{(A,I)} = \min\{[\![P[\alpha/x]]\!]^{(A,I)} \mid \alpha \in A\}$
  Errata in quanto α è una denotazione e non una connotazione! Pertanto $P[\alpha/x]$ non è ammesso dalla sintassi.
- $[\![\forall x.P]\!]^{(A,I)} = \min\{[\![P[t/x]]\!]^{(A,I)} \mid t \in Term\}$ dove Ter è l'insieme di tutte le connotazioni per termini nel nostro linguaggio.
  Errata in quanto il mio mondo potrebbe avere molte più denotazioni per termini di quelle rappresentabili sintatticamente tramite connotazioni. Esempio: $A = \mathbb{R}$ poichè l'insieme delle connotazioni è sempre enumerabile.

Per esempio: elefante è una connotazione, mentre la denotazione è quell'animale in carne ed ossa, non posso manipolare delle denotazioni in questo caso, quindi non avrebbe senso utilizzarlo.

Nel secondo caso sto testando molte più cose (connotazioni sono molti di più rispetto alle denotazioni in quanto esistono i sinonimi)

L'idea principale è tenersi una lista (una mappa) che associ nomi (connettivi) e denotazioni del mondo, questa è **l'ambiente** indicata con la lettere csi.

10.6.2 Semantica della logica primo ordine

▼ Ricorsione strutturale

Definizione di semantica classica della logica del prim'ordine.
Sia (A, I) un mondo e ξ un ambiente sul mondo. Definiamo per induzione strutturale

$[\![x]\!]^{(A,I),\xi} = \xi(x)$
$[\![f^n(t_1, \ldots, t^n)]\!]^{(A,I),\xi} = I(f^n)([\![t_1]\!]^{(A,I),\xi}, \ldots, [\![t^n]\!]^{(A,I),\xi})$

$[\![P^n(t_1, \ldots, t^n)]\!]^{(A,I),\xi} = I(P^n)([\![t_1]\!]^{(A,I),\xi}, \ldots, [\![t^n]\!]^{(A,I),\xi})$
$[\![\bot]\!]^{(A,I),\xi} = 0$
$[\![\top]\!]^{(A,I),\xi} = 1$
$[\![\neg P]\!]^{(A,I),\xi} = 1 - [\![P]\!]^{(A,I),\xi}$
$[\![P_1 \wedge P_2]\!]^{(A,I),\xi} = \min\{[\![P_1]\!]^{(A,I),\xi}, [\![P_2]\!]^{(A,I),\xi}\}$
$[\![P_1 \vee P_2]\!]^{(A,I),\xi} = \max\{[\![P_1]\!]^{(A,I),\xi}, [\![P_2]\!]^{(A,I),\xi}\}$
$[\![P_1 \Rightarrow P_2]\!]^{(A,I),\xi} = \max\{1 - [\![P_1]\!]^{(A,I),\xi}, [\![P_2]\!]^{(A,I),\xi}\}$
$[\![\forall x.P]\!]^{(A,I),\xi} = \min\{[\![P]\!]^{(A,I),\xi[x\mapsto\alpha]} \mid \alpha \in A\}$
$[\![\exists x.P]\!]^{(A,I),\xi} = \max\{[\![P]\!]^{(A,I),\xi[x\mapsto\alpha]} \mid \alpha \in A\}$

dove $\xi[x \mapsto \alpha]$ associa α a x e $\xi(y)$ a y.

10.6.3 Conseguenza logica in Primo ordine

Dobbiamo prendere in questo caso considerazione della definizione più complessa del mondo (ricordarsi che nelle logiche di ordine superiore è proprio questa ulteriore complessità che non permette di avere una completezza).

Quindi dobbiamo tenere conto del significato di **(A, I),** ξ**.**

▼ Slide

Tutte le definizioni viste per la logica proposizionale classica che facevano riferimento alle nozioni di mondo e semantica rimangono identiche per la logica del prim'ordine classica con le nuove definizioni di mondo (e ambiente) e semantica.

Esempio: $\Gamma \Vdash G$ quando in ogni mondo (A, I) e ambiente ξ si ha che se $[\![F]\!]^{(A,I),\xi} = 1$ per ogni $F \in \Gamma$ allora $[\![G]\!]^{(A,I),\xi} = 1$.

10.7 Proprietà esiste e per ogni

Queste proprietà espandono la lista CAIDANA delle proprietà presenti in Connettivi Logici, correttezza, variabili.

10.7.1 Completezza debole

▼ Slide

se

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

allora

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

(per il teorema di completezza debole)
e inoltre vi è (e sappiamo qual'è) un algoritmo f che ad ogni input x associa un output $f(x)$ tale che $P(x, f(x))$

$P(x, y)$ viene chiamata la specifica dell'algoritmo

10.7.2 Commutatività e non

▼ Slide

Quantificatori dello stesso tipo commutano:

$\forall x.\forall y.P \equiv \forall y.\forall x.P$
$\exists x.\exists y.P \equiv \exists y.\exists x.P$

Quantificatori di tipo diverso NON commutano:

$\exists x.\forall y.P \Vdash \forall y.\exists x.P$
$\forall x.\exists y.P \not\Vdash \exists y.\forall x.P$

Esempio: $\forall x.\exists y.x < y$ vs $\exists y.\forall x.x < y$ in $\mathbb{N}$.

chiaramente se Sono gli stessi x e y in due per ogni es $\forall A, \forall B$ questo è uguale a dire $\forall B, \forall A$, stessa cosa per l'esiste.

Ma il senso della frase cambia nel caso in cui abbia un per ogni e in seguito un esiste.

10.7.3 Semidistributività

In certi casi (non per tutti) posso spostare (addirittura eliminare!) i quantificatori

▼ Slide

Le seguenti equivalenze possono essere utilizzate per spostare i quantificatori in posizione di testa nelle formule:

$\forall x.(P \wedge Q) \equiv (\forall x.P) \wedge (\forall x.Q)$ (usata da dx a sx)
$\exists x.(P \vee Q) \equiv (\exists x.P) \vee (\exists x.Q)$ (usata da dx a sx)

$\forall x.P \equiv P$ se $x \notin FV(P)$ (usata da dx a sx)
$\exists x.P \equiv P$ se $x \notin FV(P)$ (usata da dx a sx)

$\forall x.(P \vee Q) \equiv (\forall x.P) \vee Q$ se $x \notin FV(Q)$ (usata da dx a sx)
$\exists x.(P \wedge Q) \equiv (\exists x.P) \wedge Q$ se $x \notin FV(Q)$ (usata da dx a sx)

10.7.4 De morgan

$\neg\forall x.P \equiv \exists x.\neg P$ solo in logica classica
$\exists x.\neg P \Vdash \neg\forall x.P$ in logica intuizionista
$\neg\exists x.P \equiv \forall x.\neg P$ in logica classica e intuizionista

Attenzione: per dimostrare che $\neg\forall x.P$ basta dimostrare che $\exists x.\neg P$ ovvero è sufficiente un controesempio. Ma per dimostrare $\neg\exists x.P$ dobbiamo dimostrare $\forall x.\neg P$ ovvero serve una dimostrazione.

10.7.5 Equivalenze notevoli

Sia $x \notin FV(Q)$ (sempre vero per un qualche Q' che sia α-convertibile con Q). Si ha

$(\forall x.P) \Rightarrow Q \equiv \exists x.(P \Rightarrow Q)$ solo in logica classica
$\exists x.(P \Rightarrow Q) \Vdash (\forall x.P) \Rightarrow Q$ in logica intuizionista
$(\exists x.P) \Rightarrow Q \equiv \forall x.(P \Rightarrow Q)$
$Q \Rightarrow (\exists x.P) \equiv \exists x.(Q \Rightarrow P)$
$Q \Rightarrow (\forall x.P) \equiv \forall x.(Q \Rightarrow P)$

▼ Altro ancora

$$
\begin{array}{lll}
 & \neg\forall x \in A.P(x) & \neg\exists x \in A.P(x) \\
= & \neg\forall x.(x \in A \Rightarrow P(x)) & = \neg\exists x.(x \in A \wedge P(x)) \\
\equiv & \neg\forall x.(\neg x \in A \vee P(x)) & \equiv \forall x.\neg(x \in A \wedge P(x)) \\
\equiv & \exists x.\neg(\neg x \in A \vee P(x)) & \equiv \forall x.(\neg x \in A \vee \neg P(x)) \\
\equiv & \exists x.(x \in A \wedge \neg P(x)) & \equiv \forall x.(x \in A \Rightarrow \neg P(x)) \\
= & \exists x \in A.\neg P(x) & = \forall x \in A.\neg P(x)
\end{array}
$$

Nota: mentre la seconda vale anche intuizionisticamente, la prima vale intuizionisticamente solo nella direzione $\exists x \in A.\neg P(x) \Vdash \neg\forall x \in A.P(x)$, compresa la sua variante $\exists x \in A.P(x) \Vdash \neg\forall x \in A.\neg P(x)$

10.8 Deduzione naturale

In questa sezione di appunto andiamo ad indagare le regole della deduzione naturale per la logica di primo ordine, per questo motivo linko la deduzione naturale in ambito proposizionale Deduzione naturale

Vogliamo utilizzare delle cosa uguali a meno di alfa conversione.

10.8.1 Introduzione Per ogni

▼ Forma generale

$$
(\forall_i)\ \frac{\begin{array}{c}\vdots\\ P[y/x]\end{array}}{\forall x.P}\quad y \notin FV(Foglie(\vdots))
$$

dove $Foglie(\vdots)$ sono le foglie non (ancora!) cancellate nel sotto-albero $\vdots$

Attenzione: una foglia può non essere ancora cancellata in $\vdots$ ma sembrare cancellata in quanto cancellata successivamente da una regola applicata successivamente.

Idea: per concludere che P vale per un x qualunque basta dimostrare che P vale per un x (qui ridenominato in y) che sia veramente qualunque ovvero sul quale non vi siano altre ipotesi ovvero tale per cui y non compaia in nessuna ipotesi.

Attento che Y non deve affatto appartenere alle variabili libere delle foglie!

Questo mi dice che devo utilizzare solamente solamente una variabile che non è stata già presa (quindi libera, data dall'esterno)

Suggerimento: per non sbagliare mai ti basterebbe prendere una variabile mai presa prima.

/to

▼ Correttezza ed invertibilità intuizionista

Correttezza intuizionista: $P[y/x] \Vdash \forall x.P$ in quanto se $p \in [\![P[y/x]]\!]$ e p non chiama funzioni di libreria "che usano y" allora la funzione $f(x) = p[x/y]$ è una funzione polimorfa che ad ogni x restituisce un $p[x/y] \in [\![P[y/x][x/y]]\!] = [\![P]\!]$ e quindi $f \in [\![\forall x.P]\!]$.

Invertibilità intuizionista: se $f \in [\![\forall x.P]\!]$ allora f è una funzione polimorfa tale che $f(x) \in [\![P]\!]$ per ogni x. In particolare $f(y) \in [\![P[y/x]]\!]$.

▼ Correttezza classica

Correttezza classica: Dimostriamo che per ogni Γ si ha che se $\Gamma \Vdash P[y/x]$ dove $y \notin FV(\Gamma)$ allora $\Gamma \Vdash \forall x.P$. Infatti sia (A, I) un mondo e ξ un environment tali per cui $[\![F]\!]^{(A,I),\xi} = 1$ per ogni $F \in \Gamma$. Per definizione di conseguenza logica si ha $[\![P[y/x]]\!]^{(A,I),\xi} = 1$. Per induzione strutturale su P si dimostra che $[\![P[y/x]]\!]^{(A,I),\xi} = [\![P]\!]^{(A,I),\xi[x\mapsto\xi(y)]} = 1$. Tale risultato deve valere non solo per ξ, ma anche per $\xi[y \mapsto \alpha]$ per ogni $\alpha \in A$ dal momento che se $[\![F]\!]^{(A,I),\xi} = 1$ allora anche $[\![F]\!]^{(A,I),\xi[y\mapsto\alpha} = 1$ per ogni $F \in \Gamma$ in quanto $y \notin FV(\Gamma)$. Si ha pertanto $\min\{[\![P]\!]^{(A,I),\xi[y\mapsto\alpha][x\mapsto\xi(y)]} \mid \alpha \in A\} = \min\{[\![P]\!]^{(A,I),\xi[x\mapsto\alpha]} \mid \alpha \in A\} = \min\{1 \mid \alpha \in A\} = 1$.

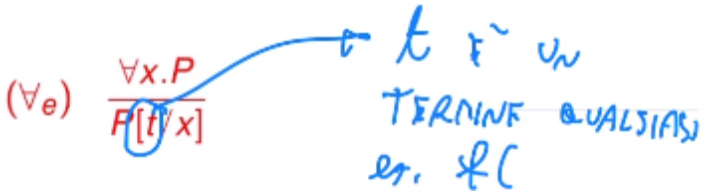
Io sintatticamente ci metto y e il mondo mi risponde xi y. Ora l'ambiente xi mi dice che x vale xi di y e quindi è la stessa. Ma lo devo dimostrare per induzione strutturale.

▼ Invertibilità classica

Invertibilità classica: $\forall x.P \Vdash P[y/x]$. Infatti se $[\![\forall x.P]\!]^{(A,I),\xi} = \min\{[\![P]\!]^{(A,I),\xi[x\mapsto\alpha]} \mid \alpha \in A\} = 1$ allora si ha $[\![P]\!]^{(A,I),\xi[x\mapsto\xi(y)]} = [\![P[y/x]]\!]^{(A,I),\xi} = 1$.

10.8.2 Eliminazione per ogni

▼ Forma generale



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

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

Scrittura informale:
. . . e quindi $\forall x.P$
e quindi $P[t/x]$

Dovrei passare per la formula più complessa a volte! A volte è più semplice dimostrare il generale che il particolare perché possiedo induzione strutturale e simili.

▼ Correttezza intuizionista e classica

Correttezza intuizionista: $\forall x.P \Vdash P[t/x] \Vdash$ in quanto se $f \in [\![\forall x.P]\!]$ allora $f(t) \in [\![P[t/x]]\!]$.

Correttezza classica: $\forall x.P \Vdash P[t/x]$ in quanto per ogni mondo (A, I) ed environment ξ tali che $[\![\forall x.P]\!]^{(A,I),\xi} = \max\{[\![P]\!]^{(A,I),\xi[x\mapsto\alpha]} = 1$ si ha $[\![P[t/x]]\!]^{(A,I),\xi} = [\![P]\!]^{(A,I),\xi[x\mapsto[\![t]\!]^{(A,I),\xi}]} = 1$ per induzione strutturale su P.

La regola è chiaramente non invertibile: per esempio, $Pari(2)$ ma non si ha $\forall x.Pari(x)$.

Anche da questo possiamo sapere che non possiamo andare a cercare tutte le variabili, sono infiniti! L'algoritmo non concluderebbe mai.

10.8.3 Introduzione Esiste

Questa dimostrazione è praticamente uguale all'eliminazione del per ogni, mentre l'eliminazione è simile all'introduzione

▼ Forma generale

Regole di introduzione:

$$(\exists_i) \quad \frac{P[t/x]}{\exists x.P}$$

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

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

Scrittura informale:
...e quindi $P[t/x]$
e quindi $\exists x.P$

10.8.4 Eliminazione dell'esiste

In questa forma io non ho nessuna informazione sulla x, non posso prendere una x che è già stata utilizzata nella conclusione C oppure nelle foglie.

Deve essere una variabile libera!, non deve avere nessuna altra ipotesi presa da altro.

▼ Forma generale

$$(\exists_e)\ \frac{\exists x.P \qquad \begin{array}{c}[P[y/x]]\\ \vdots \\ C\end{array}}{C} \qquad y \notin FV(C) \cup FV(Foglie(\vdots))$$

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.

10.9 Completezza ed incompletezza di Gode

10.9.1 Primo teorema

▼ Enunciato

Primo teorema di incompletezza di Goedel (1931)

In ogni teoria matematica Γ sufficientemente espressiva da contenere l'aritmetica, esiste una formula P tale che, se $\Gamma \nvdash \bot$ allora $\Gamma \nvdash P$ e $\Gamma \nvdash \neg P$.

Gamma voglio identificare un singolo mondo.

Se contiene l'aritmetica ho i numeri la somma i prodotti e simili. Se gamma è consistente (quindi interessante, sennò tutto e dimostrabile, è anche un modo

per dire che non ho più mondi).
Allora **non posso filtrare in maniera da tenerne solamente uno.** il gamma deve tenere anche dei mondi in più.

Quindi quando gamma parla di artimetica non si può filtrare fino a un singolo mondo.

Quindi qualunque aritmetica prendo, non posso mai filtrare fino a singolo mondo!

è una incompletezza di Gamma, ma non è una incompletezza delle regole!

▼ Abbozzo

È una delle prime assolute codifiche!

bigezione fra formule ed alberi, e la sintassi dimostrazione e poi utilizza il paradosso del mentitore (io mento) crea un numero che dice che non è dimostrabile, quindi fonde livello e metalivello per cui non può né essere dimostrabile né essere indimostrabile (altrimenti sarebbe inconsistente).
Entrambe sono false

Io sono dimostrabile se e solo se io non sono dimostrabile, quindi entrambe devono essere false.

Uno dei teoremi più storpiati dai divulgatori scientifici. Ma allo stesso tempo è uno dei teoremi più profondi della logica.

Se abbiamo abbastanza ipotesi da poter identificare solamente un singolo mondo, allora vale il concetto di terzo escluso, quindi o vale F conseguenza logica del mondo oppure not F è conseguenza logica

(nel caso contrario posso avere sia non G sia G non sono conseguenze logiche di più mondi).

Godel **trova la P**, però quel singolo P è ben conosciuto, non è molto interessante.

10.9.2 Secondo teorema

Questo ha un apporto molto maggiore, molto importante, la base dell'informatica.

▼ Enunciato

Secondo teorema di incompletezza di Goedel

Nessuna teoria Γ sufficientemente espressiva da contenere l'aritmetica e consistente (ovvero tale che $\Gamma \nvdash \bot$) è in grado di dimostrare la sua consistenza.

Non riesco mai a concludere la consitenza della logica, dovrei rimettermi al metalivello continuamente, senza finire mai.

Non possiamo **mai essere sicuri della consistenza di una teoria**, e alla fin fine la logica, la matematica si può paragonare alla religione da questo punto di vista. Noi non siamo sicuri che sia vero. Serve l'atto di fede.

11 Strutture algebriche

11.1 Differenza matematica e informatica

Una osservazione per quanto riguarda la logica intuizionista è che sta a metà fra matematica e informatica perché la dimoostrazione intuizionista possiede in sé un algoritmo e una struttura di dati.

Infatti di solito l'informatico scrive senza fare la dimostrazione dell'algoritmo mentre il matematico scrive la dimostrazione senza fare l'algoritmo (inoltre può definire degli enti ed oggetti che non siano rappresentabili come dati in quanto possono essere infiniti.

Un opinione personale è che l'informatica è più pratica, e limitata in quanto non può estendersi al ragionamento infinito ma deve essere limitata al calcolo. Questo è si molto più utile ma molto meno creativo. Si può attaccare però il problema in questi modi!

11.1.1 Osservazioni generali

La matematica utilizza questi metodi da molto tempo prima dell'esistenza di una macchina di calcolo.

I concetti generali di matematica che si sono sviluppati nei secoli sono astrazioni e generalizzazioni. (non esiste ancora una base simile in informatica, e.g. le classi di

java hanno proprio degli errori logici al suo interno, anche se non so esattamente quali, ma coen dice di sì)

11.2 Astrazione e generalizzazione

11.2.1 Tesi di Church-Turing

Questa tesi (non è un teorema) definisce il significato di **espressività di un linguaggio di programmazione**.

In altre parole deve avere:

1. Un modo di ciclare
2. branching (if condition)
3. E numeri

▼ Enunciato

> **Tesi di Church-Turing**
>
> Qualunque problema di calcolo risolvibile da un linguaggio di programmazione general-purpose lo è da qualunque altro linguaggio general-purpose.

Ma quindi i linguaggi di programmazione non cambiano per capacità di calcolo, bensì nella **capacità di astrazione e generalizzazione.**

11.2.2 Astrazione

> **Astrazione**
>
> “Processo tipico della matematica che permette di definire enti, concetti o procedure matematiche, estraendo e isolando alcune caratteristiche comuni a più oggetti e trascurandone altre”

Alcune caratteristiche le trascuro perché sono accessorie (dipendono dall'implementazione, dal caso specifico), questo mi permetti cambiare una implementazione trattenendo aspetti degli oggetti che mi interessino. esempio di astrazione è il **quozientamento** perché in questa operazione trattenevo solamente gli aspetti che mi interessavano buttando via tutto il resto. (il quozientamento non esiste in programmazione per la sua incapacità di gestire l'infinito)

▼ Esempio implementazione e astrazione di numeri naturali

Esempi di astrazione dai dettagli implementativi:

1. implementazioni concrete dei naturali in teoria degli insiemi
   $0 := \emptyset$, $1 := \{\emptyset\}$, $2 := \{\emptyset, \{\emptyset\}\}$, ..., n+1 := $n \cup \{n\}$, ...
   ove $2 + 3 = 5$ e $2 \in 5$ e $2 \subseteq 5$
   oppure
   $0 := \emptyset$, $1 := \{\emptyset\}$, $2 := \{\{\emptyset\}\}$, ..., n+1 := $\{n\}$, ...
   ove $2 + 3 = 5$ e $2 \notin 5$ e $2 \not\subseteq 5$
   vs concetto astratto di numero naturale

$$\exists \mathbb{N}, O, S.$$
$$\forall n.(n \in \mathbb{N} \iff n = O \vee \exists m.(m \in \mathbb{N} \wedge n = Sm)) \wedge$$
$$(\forall m.O \neq Sm) \wedge$$
$$(\forall n, m.Sn = Sm \Rightarrow n = m)$$

L'implementazione posso farla in molti modi, qui ne sono rappresentati due.

Mentre il concetto astratto sono le regole che mi definiscono l'ente (in questo caso gli **assiomi di peano,** il fatto che devo avere uno Zero una successione e l'insieme generali dei numeri natuali, un elemento appartiene a questo insieme solamente se è uguale allo zero oppure è un successore di un numero di questo insieme. S deve essere iniettiva. E nessun successore è uguale allo zero).

▼ Esempio implementazione e astazione liste di interi

Esempi di astrazione dai dettagli implementativi:

1. implementazioni concrete di liste di interi

```
struct node {int item; struct node* next;}
```

oppure

```
struct node {int item; struct node* next;
                       struct node* prev;}
```

vs concetto astratto (interfaccia) di una lista

```
node* empty();
insert(int n, node* l);
remove(int n, node* l);
...
```

11.2.3 Generalizzazione

Generalizzazione

"Generalizzare significa prendere una definizione o una relazione che si applica a certi casi, e definirla in un nuovo modo tale che nei casi di prima dia gli stessi risultati, però si applichi anche ad altri casi."

Quindi vorrei che una caratteristica in un caso particolare sia vero anche in molti altri casi!

▼ Benefici di ciò (4)

1. Riuso: le buone generalizzazioni si applicano moltissime volte
2. Chiarezza: le generalizzazioni catturano concetti di alto livello, introducono nomi comprensibili, chiarificano e minimizzano le assunzioni
3. Decoupling: attraverso astrazione e generalizzazione è possibile fare in modo che la correttezza di una parte non dipenda dall'implementazione di un'altra parte, permettendo modifiche indipendenti
4. Correttezza: l'alternativa al riuso è il cut&paste che può propagare errori, da correggere poi in tutte le copie, e introdurre errori quando il codice viene riusato ove non valgano gli stessi invarianti sui dati

11.3 Struttura algebrica

11.3.1 L'elemento neutro (generalizzazione di essa)

Teorema T1: $\forall e \in \mathbb{N}.((\forall x.x + e = x) \Rightarrow e = 0)$
Dimostrazione: sia $e \in \mathbb{N}$ t.c. $\forall x.x + e = x$ (H). Per H, $0 + e = 0$. Poichè $e = 0 + e$ si ha $e = 0$. □

Teorema T2: $\forall e \in \mathbb{Z}.((\forall x.e * x = x) \Rightarrow e = 1)$
Dimostrazione: sia $e \in \mathbb{Z}$ t.c. $\forall x.e * x = x$ (H). Per H, $e * 1 = 1$. Poichè $e = e * 1$ si ha $e = 1$. □

Gli enunciati e le prove sono simili, ma differenti, così come i tipi di dato e le operazioni. C'è una generalizzazione comune? È istruttivo studiarla?

Si una generalizzazione esiste ed è questa:

Teorema G:
$\forall \mathbb{A}.\forall \circ \in \mathbb{A}^{\mathbb{A}\times\mathbb{A}}.\forall e_r, e_l \in \mathbb{A}.$
$(\forall x.x \circ e_r = x) \wedge (\forall x.e_l \circ x = x) \Rightarrow$
$e_l = e_r$
Dimostrazione: siano $\mathbb{A}$ insieme, $\circ \in \mathbb{A}^{\mathbb{A}\times\mathbb{A}}$ e $e_l, e_r \in \mathbb{A}$ t.c. $\forall x.x \circ e_r = x$ (H1) e $\forall x.e_l \circ x = x$ (H2). Per H1 e H2 si ha $e_l = e_l \circ e_r = e_r$. □

T1 è G nel caso particolare $\mathbb{A} = \mathbb{N}$, $\circ = +$, $e_l = 0$.
T2 è G nel caso particolare $\mathbb{A} = \mathbb{Z}$, $\circ = *$, $e_r = 1$.
Altri esempi di istanze di G:

- $\forall e \in \mathbb{N}.((\forall x.e + x = x) \Rightarrow e = 0)$
- $\forall e \in \mathbb{B}.((\forall b.e \,\&\, b = e) \Rightarrow b = true)$

G è una generalizzazione utile di T1 e T2 (ha altre istanze oltre a T1 e T2)

Queste generalizzazioni sono utili nel caso mi creino una **teoria** utile da studiare dal punto di vista astratto (così posso scoprire altre proprietà).

Teorema G:
$\forall \mathbb{A}.\forall \circ \in \mathbb{A}^{\mathbb{A}\times\mathbb{A}}.\forall e_r, e_l \in \mathbb{A}.$
$(\forall x.x \circ e_r = x) \wedge (\forall x.e_l \circ x = x) \Rightarrow$
$e_l = e_r$

Introduciamo due definizioni:

1. e è un elemento neutro a sinistra per $\circ$ sse $\forall x.e \circ x = x$
2. e è un elemento neutro a destra per $\circ$ sse $\forall x.x \circ e = x$
3. e è un elemento neutro per $\circ$ sse è neutro sia a sinistra che a destra

Teorema G: se un'operazione binaria su $\mathbb{A}$ ha sia un elemento neutro a sinistra che uno a destra, allora questi coincidono.

G ha un enunciato più comprensibile

Se questa teoria è utile per comprendere concetti più bassi (dal punto di vista di livelli di astrazione) allora posso dire che queste sono teorie **informative**.

11.3.2 Definizione struttura algebrica

Il **magma** è il concetto fondamentale per una struttura algebrica come in definizione.

▼ Definizione

Una struttura algebrica è una tupla (o ennupla) formata da uno o più insiemi, zero o più elementi, zero o più funzioni (chiamate operazioni) su tali insiemi e zero o più assiomi (= proprietà) che devono essere soddisfatte.

Esempi:

- un left unital magma è una tripla $(\mathbb{A}, \circ, e)$ t.c. $\circ \in \mathbb{A}^{\mathbb{A}\times\mathbb{A}}$ e e è un elemento neutro a sinistra per $\circ$.
- un right unital magma è una tripla $(\mathbb{A}, \circ, e)$ t.c. $\circ \in \mathbb{A}^{\mathbb{A}\times\mathbb{A}}$ e e è un elemento neutro a destra per $\circ$.
- un unital magma è una tripla $(\mathbb{A}, \circ, e)$ t.c. $\circ \in \mathbb{A}^{\mathbb{A}\times\mathbb{A}}$ t.c. $(\mathbb{A}, \circ, e)$ è sia un left unital magma che un right unital magma

Ce ne sono centinaia, ne introdurremo alcuni. Interi scaffali di librerie sono dedicate allo studio di una sola di queste strutture algebriche!

Chiamiamo magma perché non è ancora solidificato, qualcosa di abbastanza astratto.

Dopo avere introdotto questi concetti astratti posso studiarle! Posso studiare una cosa che ho creato, mi piace sta scienza, perché chi ha creato qualcosa non sa a priori bene cosa ha creato.

▼ Prodotto cartesiano in left unital magma

Una volta introdotte, le strutture algebriche stesse diventano oggetto di studio. Esempio:

Teorema (prodotto cartesiano di left unital magmas): siano $(\mathbb{A}, \circ, a)$ e $(\mathbb{B}, \bullet, b)$ due left unital magma. Allora anche $(\mathbb{A}, \circ, a) \times (\mathbb{B}, \bullet, b) := (\mathbb{A} \times \mathbb{B}, \oplus, \langle a, b \rangle)$, chiamato prodotto cartesiano dei due left unital magmas, è un left unital magma dove $\langle x_1, y_1 \rangle \oplus \langle x_2, y_b \rangle := \langle x_1 \circ x_2, y_1 \bullet y_2 \rangle$.
Dimostrazione: sia $c \in \mathbb{A} \times \mathbb{B}$, ovvero $c = \langle x, y \rangle$ e dimostriamo $\langle a, b \rangle \oplus \langle x, y \rangle = \langle x, y \rangle$. Si ha
$\langle a, b \rangle \oplus \langle x, y \rangle = \langle a \circ x, b \bullet y \rangle = \langle x, y \rangle$. □

Esempio di istanza: poichè $(\mathbb{R}, +, 0)$ è un left unital magma, allora lo è anche $(\mathbb{C}, +, 0)$.

Nell'esempio C è un prodotto cartesiano di R per questo motivo riesco a dirlo da questa generalizzazione!

11.3.3 In programmazione

11.4 Morfismi

Il morfismo mi permette di **preservare** alcune proprietà che perdo in astrazione's trasformazione (un esempio è la perdita della struttura di numeri quando lo rappresento con un LUM (Left Unital Magma)

▼ Esempio

Esempio: siano $(A, \circ, a)$ e $(B, \bullet, b)$ due left unital magma. Un morfismo da $(A, \circ, a)$ a $(B, \bullet, b)$ è una funzione $f \in B^A$ t.c.

1. $f(a) = b$
2. $\forall x, y. f(x \circ y) = f(x) \bullet f(y)$

Esempio: $(\mathbb{N}, +, 0)$ e $(\mathbb{N}, *, 1)$ sono due left unital magma. La funzione $f(n) = 2^n$ è un morfismo dal primo al secondo in quanto:

1. $2^0 = 1$
2. $\forall x, y. 2^{(x+y)} = 2^x * 2^y$

Questo concetto è importante anche in informatica perché vuol dire che posso prima applicare la funzione sulla somma di implementazioni oppure applicarli singolarmente.

11.4.1 Isomorfismo

L'isomorfismo è simile a un morfismo a due parti (in cui vale questa relazione in entrambe le parti) quindi **funzione bigettiva** in cui anche l'inverso sia un morfismo. Questa cosa mi dice che esiste una certa **corrispondenza fra strutture algebriche**.

▼ Enunciato ed esempio

Un isomorfismo di due strutture algebriche è un morfismo dalla prima alla seconda che sia una funzione biettiva la cui inversa sia anch'essa un morfismo.

Due strutture algebriche sono isomorfe se c'è almeno un isomorfismo fra di esse.

Tutte le manipolazioni su una struttura possono essere effettuate su quella isomorfa e viceversa, a seconda di cosa sia più conveniente.

Esempio: $(\mathbb{R}_0^+, +, 0)$ e $(\mathbb{R}^+, *, 1)$ sono isomorfe come testimoniato dal morfismo $e^{\cdot}$ e dal suo morfismo inverso $\log \cdot$. Invece di moltiplicare numeri grandi posso passare alla scala logaritmica e sommare numeri piccoli.