Logica Matematica

Calabrigo Massimo

February 18, 2020

Contents

1 Linguaggio Proposizionale

1.1 Cos'è il linguaggio proposizionale - Formule

Il linguaggio proposizionale è un linguaggio, ovvero l'insieme di tutte le possibili combinazioni di formule proposizionali. Il linguaggio proposizionale è formato da:

- Connettori: $'\wedge','\vee','\neg','\rightarrow'$
- elementi sintattici: $'(',')'$
- lettere proposizionali: $'p','q', ...$

Priorità dei connettivi logici:

1. $\neg(A)$
2. $A \wedge B$
3. $A \vee B$
4. $A \rightarrow B$

Una formula può essere:

- negazione: $\neg(a)$
- congiunzione:$a \wedge b$
- disgiunzione:$a \vee b$
- implicazione:$a \rightarrow b$

Una **sottoformula** è una sottostringa di una formula, che è anch'essa una formula.

1.2 Equivalenza e Conseguenza Logica

Due formule A e B sono logicamente equivalenti $A \equiv B$ quando A e B hanno gli stessi output.
B è conseguenza logica di A,$A \vdash B$, quando se A è vero anche B deve essere vero, e se B è falso, anche A deve essere falso.
Esempio: $p \wedge q \vdash p = G \vdash F$

F	F	F	F
F	F	F	V
F	V	V	F
V	V	V	V

1.3 DeMorgan

- $\neg(F \wedge G) = \neg(F) \vee \neg(G)$
- $F \rightarrow G = \neg(F) \vee G$
- $F \wedge (G \vee H) = (F \wedge G) \vee (F \wedge H)$

1.4 Validità, Soddisfacibilità e insoddisfacibilità

Detta F una formula:

1. F è valida se v(F) = vero, per ogni valutazione v
2. F è soddisfacibile se v(F) = vero, per almeno una valutazione v
3. F è insoddisfacibile se v(F) = falso, per ogni valutazione v

Lemma: Siano F e G formule:

- se $F \vdash G$, allora $F \rightarrow G$ è valida
- se $F\neg \vdash G$, allora $F \wedge \neg(G)$ è soddisfacibile

N.B: Vediamo che per verificare la conseguenza logica è necessaria la validità, mentre per confutarla è sufficiente la soddisfacibilità.
Per i tableaux varrà la stessa cosa. Al posto di cercare se $F \vdash G$ è valido, possiamo cercare se $F\neg \vdash G$ è insoddisfacibile, se fosse soddisfacibile, sapremmo che $F \vdash G$ non potrebbe essere valida.

1.5 FNC e FND

Teo: Una qualsiasi formula F può essere trasformata in una funzione in Forma Normale Congiuntiva (G1) e/o in una funzione in Forma Normale Disgiuntiva (G2).
Una formula in FNC è composta da un insieme di sottoformule $(H_1, H_2, ..., H_n)$ messe tutte in congiunzione $\wedge$ tra loro. Ogni formula H_i, invece, è formata da lettere proposizionali (tipo variabili booleane) messe tutte in disgiunzione $\vee$ tra loro.
Esempio: K = $(p \vee t) \wedge (p \vee n)$, dove K è in FNC.
Scambiando gli or con gli and ottengo una formula in Forma Normale Disgiuntiva.
S = ((p and t) or (p and n)), dove S è in FND.
N.B: le formule K e S non sono le stesse, una volta ottenuta una formula FNC K non basta scambiare and e or per ottenere una formula FND S.

1.6 Algoritmo di FNC e FND

L'algoritmo serve a trasformare una qualunque funzione F in una funzione K in FNC o FND. Lo scopo è semplificare lo studio della formula e avere certi vantaggi. Per esempio in una sottofunzione di una funzione FND sappiamo che se ci sono due lettere proposizionali complementari, quella formula darà sempre falso. Esempio: $((p \wedge \neg(p) \wedge q \wedge z \wedge \neg(i) \wedge j) \vee (...))$, sappiamo che la prima sottoformula è falsa.

L'algoritmo (per FNC) esegue un numero di passi sulla funzione in input F, e smette quando K = FNC(F), altrimenti eseguo l'algoritmo su la sua prima sottoformula G. In questo caso so che G non è un letterale, quindi posso procedere in questi modi:

- se G è una doppia negazione, allora rimpiazzo G con il suo ridotto (se G = not(not(N))), allora rimpiazzo G con N
- se G è una beta-formula, rimpiazzo G con i suoi ridotti
- se G è una alpha-formula, ovvero visto che siamo in FNC se mi trovo in questa situazione: $< [A \wedge B], [C] >$, uso DeMorgan (vedi sotto).

Esempio:

$$< [(A \wedge B)], [C] >$$
$$((A \wedge B) \vee 0) \wedge C$$
$$(C \vee (A \wedge B)) \wedge (C \vee 0)$$
$$(C \vee (A \wedge B))$$
$$(C \vee A) \wedge (C \vee B)$$
$$< [A, C], [A, B] >$$

Alpha-formule:

Alpha-formule	ridotti
$F \wedge G$	F, G
$\neg(F \vee G)$	$\neg(F), \neg(G)$
$\neg(F \to G)$	$F, \neg(G)$

Beta-formule:

Beta-formule	ridotti
$F \vee G$	F, G
$\neg(F \wedge G)$	$\neg(F), \neg(G)$
$F \to G$	$\neg(F), G$

Introduciamo una nuova notazione per l'algoritmo:

- $[p, q, w] = p \vee q \vee w$
- $< p, q > = p \wedge q$

Esempio: $((p \vee t) \wedge (p \vee n))$ diventa $< [p, t], [p, n] >$.

Vediamo un esempio significativo di applicazione dell'FNC:

INIZIO

1. **F = '**$((r \wedge \neg(s)) \vee \neg(p \to q))$**'**
   Per prima cosa cambiamo la sintassi della funzione per usare l'algoritmo, qui potrei decidere di trovare $< [..] >$, e in questo caso troverei la FNC, oppure $[< .. >]$, per trovare la FND. Noi scegliamo la prima:

2. ‘$< [((r \wedge \neg(s)) \vee \neg(p \rightarrow q))] >$‘
   Abbiamo due grandi sottoformule: $G = (r \wedge \neg(s))$ e $H = \neg(p \rightarrow q)$. G e H sono messe in $\vee$, questo corrisponde alla prima beta-formula, quindi riscriviamo i ridotti:

3. ‘$< [((r \wedge \neg(s)), \neg(p \rightarrow q))] >$‘
   Ora guardiamo la sottoformula a $G = ((r \wedge \neg(s))$. Corrisponde alla prima alpha-formula F and G dove F = r e G = $\neg(s)$. Ora viene un passaggio bastardo, perchè non basta sostituire i ridotti ma bisogna usare la formula di DeMorgan sovracitata $(F \vee (G \wedge H)) = ((F \vee G) \wedge (F \vee H))$, dove F = $\neg(p \rightarrow q)$, G = r, e H = $\neg(s)$. Quindi riscriviamo:

4. ‘$< [\neg(p \rightarrow q), r], [\neg(p \rightarrow q), \neg(s)] >$‘
   Rimaniamo un attimo su questo passaggio, perchè lo abbiamo fatto e non abbiamo solo scritto i ridotti? Beh noi abbiamo $< [((r \wedge \neg(s)), \neg(p \rightarrow q))] >$, utilizzando gli assegnamenti per F,G e H di prima avremmo: $< [(G \wedge H), F] >$, possiamo ignorare l'and <>, e quindi, se sostituiamo le [,] otteniamo: $(G \wedge H) \vee F$. Ora se sotituissimo $(G \wedge H)$ con i ridotti nella alpha-formula otterrei $(G, H) \vee F$. Ora sostituiamo i valori a G e H e rimettiamo le parentesi: $(G, H) \vee F$ diventa $(r \vee \neg(s)) \vee \neg(p \rightarrow q)$, che diventa $< [(r \vee \neg(s)), \neg(p \rightarrow q)] >$. Abbiamo ottenuto una formula sbagliata perchè $< [(r \vee \neg(s)), \neg(p \rightarrow q)] >$ è diversa da $< [(rand \neg(s)), \neg(p \rightarrow q)] >$. Questo dimostra che se sto cercando una FNC, allora quando applico le alpha-formule devo usare anche la formula di De Morgan, se sto cercando le FND, invece, funziona al contrario (alpha-formule solo ridotti, e beta-formule ridotti + De Morgan).
   A questo punto possiamo continuare, abbiamo una alpha-formula $[\neg(p \rightarrow q)] \vee r$, ridotta diventa $(p \wedge \neg(q)) \vee r$, e poi applichiamo DeMorgan:

5. ‘$< [p, r], [\neg(q), r], [not(p \rightarrow q), \neg(s)] >$‘
   A questo punto abbiamo la stessa alpha-formula, con $\neg(p \rightarrow q) \vee \neg(s)$ che diventa $(p \wedge \neg(q)) \vee \neg(s)$:

6. ‘$< [p, r], [\neg(q), r], [p, \neg(s)], [\neg(q), \neg(s)] >$‘
   Bene abbiamo ottenuto un K che sia in FNC, dove ogni disgiunto di ogni congiunto è un letterale. Ora togliamo la sintassi scomoda e ab-

biamo finito:

7. $\mathbf{K} = \text{`}((p \vee r) \wedge (\neg(q) \vee r) \wedge (p \vee \neg(s)) \wedge (\neg(q) \vee \neg(s)))\text{`}$

FINITO

1.7 Rango di una Funzione

Il rango di una funzione F, rg(F), esprime una misura di complessità di F:

- se F è un letterale, $rg(F) = 1$
- se F è una doppia negazione $F = \neg(\neg(G))$, allora $rg(F) = rg(G) + 1$
- se F è una alpha o beta-formula $F = (G \vee H)$, $rg(F) = rg(G) + rg(H) + 1$

2 Tableaux

2.1 Cos'è un tableaux e algoritmo

Il tableau è un albero i cui nodi sono formule, ed è costituito per gradi $t_1, ..., t_n$, dove ogni grado aggiunge uno o due nodi. Il tableau segue le regole dell'FNC e FND per scomporre una formula in letterali. $E(n)$ dove n è un nodo indica l'insieme che etichetta n. Alla fine di un tableau, ovvero quando si è scomposto in letterali, bisogna dire se le varie formule scomposte siano impossibili o compatibili, e in questo ultimo caso, anche dire per che valori siano compatibili.
Lemma: Se un albero binario è infinito, allora ha un ramo infinito.
Teorema: L'algoritmo di costruzione dei tableau ha terminazione forte, ovvero termina sempre.
Un tableau è chiuso se ha letterali complementari in tutte le sue foglie, altrimenti è aperto.
Se un tableau è chiuso, F è insoddisfacibile, e viceversa.
L'ALGORITMO Quando si risolve un tableaux, lo si fa usando l'algoritmo della FNC. Ogni volta che ci troviamo in una foglia di un tableaux, possiamo:

- se la foglia è una congiunzione di disgiunti elementari $(s \vee k) \wedge (s \vee n)$, allora se non ci sono complementari $w \wedge (q \vee a)$, la foglia è aperta, altrimenti se ci sono complementari $(w \wedge \neg(w)$, la foglia è chiusa.

- se la foglia è una congiunzione di disgiunti non elementari $(d \to k \vee k) \wedge (s \to n \vee n)$, e se la foglia è una beta-formula, allora l'albero ha un nodo, se la foglia è una alpha-formula, l'albero ha 2 nodi.

I tableaux si possono usare per verificare l'equivalenza logica:
Se abbiamo $F \vdash G$, per vedere se è valida dobbiamo dimostrare $F \to G$ sia valida, ma con i tableaux posso solo verificare la Soddisfacibilità/insoddisfacibilità, quindi, posso verificare che $\neg(F \to G) = F \wedge \neg(G)$ sia insoddisfacibile.
Quindi dovrò verificare se $F, \neg(G)$ in FNC, sia insoddisfacibile oppure soddisfacibile.

2.2 Insiemi di Hintikka

Un insieme di hintikka è un insieme i cui elementi sono le funzioni di un ramo di un tableaux, con la condizione che questi sia aperto e che rispetti certe condizioni:

1. se H è chiuso (contiene letterali complementari), H non è un insieme di hintikka.
2. se $G \in H$, allora H è una doppia negazione con ridotto G.
3. se $G \in H$ è una alpha-formula, allora $G_0 \in H$ e $G_1 \in H$.
4. se $G \in H$ è una beta-formula, allora $G_0 \in H$ oppure $G_1 \in H$ (è possibile che sia G_0, che G_1 appartengano ad H).

Esempio: Date $F_0 = \neg(p \to q \vee \neg(r))$ e $F_1 = \neg(p) \vee (r \to q)$, trova H_0 e H_1:
$H_0 = \{\neg(p \to q \vee \neg(r)), \neg(\neg(p)), \neg(q), \neg(\neg(r)), p, r\}$
$H_1 = \{\neg(p) \vee (r \to q), r \to q, q\}$

Lemma: Ogni insieme di Hintikka è soddisfacibile. (perchè? perchè non hai letterali complementari).
Per dimostrarlo prendiamo come esempio H_1, creaiamo una valutazione v tale che:

- v(p) = vero se $p \in H_1$, dove p sono i letterali in H_1
- v(p) = falso se $p \notin H_1$

allora $\forall$ formula $G \in H_1$, abbiamo v(G) = vero, di conseguenza H_1 è soddisfacibile.

Come esempio prova ad applicare la valutazione v: v(p)=vero, v(r)=falso, v(q)=falso; all'insieme H_1, il risultato sarà che ogni formula $G \in H_1$ è vera.
Lemma: Se r è un ramo aperto in un tableau, allora l'unione di tutte le funzioni dei nodi figli di r, compreso r, sono un insieme di hintikka.

In breve, cos'è un insieme di Hintikka?
E' un insieme, un'unione di funzioni, che sono tutte soddisfacibili. Viene usato in combo con i tableau, dai quali si possono creare insiemi di Hintikka a patto che i tableau siano aperti; gli insiemi di Hintikka si possono creare anche da rami di tableau, ma devono essere aperti.

3 Deduzione naturale

3.1 Cos'è la deduzione naturale?

La deduzione naturale è una tecnica utilizzata per trovare la conseguenza logica tra due formule A e B, e nella deduzione si scrive come $A \triangleright B$, oppure A deduce B.
Una tecnica utile per risolvere esercizi della deduzione naturale, in una formula $A \triangleright B$ è quella di procedere dalla tesi B alle ipotesi A, in modo da capire di cosa si ha bisogno per arrivare alla fine, e "ricorsivamente" costruire l'albero, ovviamente bisogna tener conto anche delle ipotesi, e quindi il modo giusto di risolvere gli esercizi è:

1. Guardare le ipotesi iniziali e tenerle a mente
2. Guardare la tesi e vedere come si può scomporre al livello sovrastante
3. Continuare a scomporre le ipotesi finche si riesce, e quando ci si sente sicuri provare a sviluppare le ipotesi per ottenere le parti scomposte della tesi
4. Aggiungere, se servono, ipotesi che verranno successivamente scaricate, in modo da ottenere le parti scomposte della tesi, e quando si deve decidere che ipotesi aggiungere, chiedersi sempre se poi queste ipotesi potranno essere scaricate.
5. Fare in ogni momento un **elenco delle ipotesi da scaricare**. Se si riesce arrivare alla fine dell'esercizio con "ipotesi infinite", e poi fare un elenco di tutte le ipotesi e vedere se l'esercizio può funzionare, o ha bisogno di modifiche.
6. Finito

3.2 Formule della deduzione naturale

Formule normali

1. $F, G | F \wedge G$; ($\wedge$ i)
2. $F \wedge G | F$; ($\wedge$ e.1)
3. $F \wedge G | G$; ($\wedge$ e.2)
4. $F | F \vee G$; ($\vee$ i.1)
5. $G | F \vee G$; ($\vee$ i.2)
6. $T, [F] \triangleright G | F \rightarrow G$; ($\rightarrow$ i)
7. $F, F \rightarrow G | G$; ($\rightarrow$ e)
8. $F, \neg F | \perp$ ($\neg$ e)
9. $[F], [G] \triangleright (F \wedge G, H, H) | H$; ($\vee$ e)
10. $[F] \triangleright \perp | \neg F$; ($\neg$ i)
11. $\neg\neg F | F$; ($\neg\neg$ e)

Formule speciali

1. $T \triangleright \perp | F$
2. $[F \vee \neg F]$
3. $T, [F] \triangleright \perp | \neg F$
4. $F \rightarrow G, \neg G | \neg F$

3.3 Dettagli utili per la deduzione naturale, e cose poco intuitive

1. Quando sono in una situazione di $(i \Rightarrow)$, dove normalmente la regola sarebbe
$$T, F \triangleright G | (F \rightarrow G)$$
se ho già scaricato F da un altra parte allora posso fare direttamente
$$G | (F \rightarrow G)$$
senza rifare lo scaricamento (ma devo comunque segnare affianco all linea tra $G | G \rightarrow F$, che sto facendo uno scaricamento mettendo il solito numeretto).

2. Quando sono in una situazione di **ex-falso**, con una formula del tipo

$$T \triangleright \bot | F$$

significa che avendo un falso, da esso posso dedurre qualsiasi formula.

3. Nella regola ($\vee$ e), si può usare come primo campo $For\neg F$, che poi verrà automaticamente scaricato.

4 Linguaggio Predicativo

4.1 Variabile Libera

Una variabile libera è una variabile che non è legata ai quantificatori ($\exists$ e $\forall$).
Quindi, se abbiamo una formula del tipo $\forall x(r(x, y))$, allora y sarà una variabile libera, mentre x sarà una variabile legata. Possiamo pensare più facilmente ad y come una variabile vera e propria, alla quale posso sostituire qualsiasi valore io voglia (presente nel dominio), mentre per quanto riguarda la x, è una valore al quale dobbiamo sostituire, uno alla volta, tutti i valori del dominio.

4.2 Sostituzione di termini e/o formule

La scrittura $r(f(x))\{x/t\} = r(f(t))$, significa che devi sostituire alla variabile x, il termine t.
La scrittura $r(f(x))\{x/f(x)\} = r(f(f(x)))$, significa che devo sostituire alla variabile x, il termine f(x).
Per poter fare una sostituzione, devo essere sicuro che la variabile che sostituisco sia una variabile libera, inoltre devo accertarmi che, dopo averla sostituita, essa rimanga ancora libera.Esempio:

$$\forall x(r(x, y))\{x/w\}$$

Nella formula sovrastante ci sono 2 variabili: x e y. x è legata (dal quantificatore $\forall x$) mentre y è libera, e visto che io sto cercando di sostituire w ad x, e che dopo aver sostituito w, quest'ultima sarà comunque una variabile libera, poichè non è legata a nessun quantificatore, allora posso effettuare la sostituzione.

4.3 Interpretazione di una formula

Cos'è un linguaggio?
Un linguaggio è composto da simboli di costante, di relazione e di funzione, oltre che a quantificatori e connettivi logici.
Un linguaggio può anche essere definito come insieme di formule.
Cos'è l'interpretazione di una formula?
L'interpretazione è il valore che assume quella formula, al variare dello stato (vedi dopo) a cui è associata. Si scrive I, σ soddisfa F, dove F è la formula e σ è lo stato.
Per trovare l'interpretazione di una formula, dobbiamo avere l'interpretazione di un linguaggio, per esempio se dovessimo avere un linguaggio L_0, con 1 simbolo di costante c, un simbolo di relazione binario r e un simbolo di funzione f, una interpretazione possibile potrebbe essere:

- $D = \{0, 1, 2\}$
- $c^I = 1$
- $f(0) = 1, f(1) = 2, f(2) = 3$
- $r^I = \{(0, 0), (1, 2), (2, 2)\}$

4.4 Stato di una formula

Lo stato associato ad una formula, è una funzione che manda da una qualsiasi variabile, ad un elemento del dominio: $\sigma : Var \to D^I$.
Lo stato si scrive in coppia con l'interpretazione, e li si usano per vedere se una determinata formula F, possa far parte dell'insieme delle formule T del linguaggio L_0. Si scrive I, σ soddisfa F. E si legge l'interpretazione con stato sigma, soddisfa F.
Uno stato possibile potrebbe essere:

- $\sigma(x) = 0$
- $\sigma(y) = 1$
- $\sigma(w) = 2$ (con $w \neq x, y$)

Il sigma può essere usato su qualsiasi termine, ma avrà effetti diversi:

- (termine) $\sigma(variabile) \to \sigma(x)$, per i $\sigma(Var)$ definiti nello stato (esempio sopra)

- (termine) $\sigma(costante) \rightarrow \sigma(c^I)$
- (termine) $\sigma(simbolodifunzione) \rightarrow f(\sigma(t_1), \sigma(t_2), ..., \sigma(t_n))$
- (formula) $\sigma(simbolodirelazione) \rightarrow$ risolvo i sigma, e vedo se $p(0,1,3)$ appartiene a $p^I\{(1,2,3),(3,6,8),...,(2,8,34)\}$. Se appartiene allora I, σ soddisfa $p(0,1,3)$, altrimenti no.
- (formula) Per tutti i tipi di formule composte da connettivi logici, tranne i quantificatori, il soddisfa funziona in modo analogo alla soffisfazione del connettivo scelto, per esempio se ho I, σ soddisfa F or G, deve valere I, σ soddisfa F oppure I, σ soddisfa G.
- (formula) Per le formule con un quantificatore devo fare:
  - Quantificatore Esistenziale: $D^I = \{0,1,2\}$ $(\forall x(r(x,y)))$, devo controllare che I, σ soddisfi $r(x,y)$, per almeno un valore di x, quindi devo verificare:
    1. $I, \sigma[x/0]$ soddisfa $r(x,y)$ oppure
    2. $I, \sigma[x/1]$ soddisfa $r(x,y)$ oppure
    3. $I, \sigma[x/2]$ soddisfa $r(x,y)$
  - Quantificatore Per ogni: $D^I = \{0,1,2\}$ $(\forall x(r(x,y)))$, devo controllare che I, σ soddisfi $r(x,y)$, per tutti i valori di x, quindi devo verificare:
    1. $I, \sigma[x/0]$ soddisfa $r(x,y)$ e
    2. $I, \sigma[x/1]$ soddisfa $r(x,y)$ e
    3. $I, \sigma[x/2]$ soddisfa $r(x,y)$

4.5 Metodologia da seguire per l'intepretazione di una formula

Dati Una formula F, uno stato σ e un'interpretazione I, bisogna trovare se I, σ soddisfi F. Vedi i punti sovrastanti, c'è scritto cosa fare ad ogni passo!

4.6 Equivalenza e Conseguenza Logica

Due formule A e B sono logicamente equivalenti $A \equiv B$, se per ogni interpretazione I, e ogni stato σ: I, σ soddisfa A se e solo se I, σ soddisfa B.
Date 2 formule A e B, B è conseguenza logica di A se per ogni interpretazione I, e ogni stato σ: I, σ soddisfa A si ha I, σ soddisfa B