LOGICA E ALGEBRA

Alessandra Cherubini,
Stefania Adami,
Claudia Nuccio,
Emanuele Rodaro

Introduzione.

Lo studio della logica, come studio degli schemi di ragionamento, risale ai Greci ed in particolare al famoso sillogismo di Aristotele : Socrate è un uomo; ogni uomo è mortale e quindi Socrate è mortale. La logica aristotelica usa il linguaggio naturale e quindi porta in sé le ambiguità proprie di tale linguaggio. Nel 17° secolo qualche tentativo di Leibnitz ed altri di introdurre un linguaggio formale per il "calcolo logico" venne poco considerato e solo nella seconda metà dell'800 ad opera di Boole e di De Morgan, ai quali sostanzialmente si deve quella che oggi chiamiamo logica proposizionale, la logica diventa uno strumento per lo sviluppo rigoroso e formale della matematica con un proprio linguaggio formale.

In logica formale l'interesse continua ad essere focalizzato sulla verità o falsità di affermazioni e su come la verità o falsità di una affermazione possa essere determinata a partire da altre affermazioni, ma tutto questo viene fatto senza lavorare su affermazioni specifiche bensì usando simboli per rappresentare affermazioni generiche in modo da un lato di avere dei procedimenti generali che possano essere utilizzati in casi diversi, e dall'altro di evitare, attraverso la formalizzazione, errori che possono dipendere da ambiguità o sottointesi del linguaggio naturale.

Perché allora la logica può interessare all'informatica?

Logica ed informatica sono simili: si occupano entrambe di problemi di formalizzazione, elaborazione e comunicazione della conoscenza, entrambe hanno bisogno di usare un linguaggio formale.

"L'informatica è nata dalla logica come Minerva dalla testa di Giove" (G.Longo), per esemplificare questa affermazione basta notare che la teoria della computabilità nasce negli anni 30 (Gödel, Church, Kleene, Turing…), cioè ben prima dei moderni calcolatori e che la distinzione fra hardware e software, presente per la prima volta nelle architetture di calcolo disegnate da Von Neumann e Turing nel dopoguerra, era stata precedentemente proposta da Turing per l'analisi logica della deduzione.

La logica ha innumerevoli applicazioni in informatica: progettazione dei circuiti digitali, modellizzazione di macchine astratte, verifica della correttezza dei programmi, verifica delle specifiche e della correttezza di sistemi critici, data base relazionali, implementazione di protocolli di comunicazione, ingegneria della conoscenza etc. Ognuna di questa applicazioni richiede una specifica logica e così non possiamo parlare di logica ma di logiche:

- logica proposizionale,
- logica del I ordine o predicativa,
- logica modale,
- logica temporale,
- logica probalistica,
- logica fuzzy,
- logiche descrittive,
- logiche del II ordine,
- …..

e lo sviluppo di queste logiche è spesso dovuto alle applicazioni, dunque si può concludere che la logica è la matematica per l'informatica o in altre parole (come detto anche da Vardi) che

logica : informatica = analisi mat : fisica

In questo corso ci occuperemo solo delle logiche proposizionale e predicativa e di alcuni concetti fondamentali di algebra, perché algebra e logica sono pure strettamente collegate.

L'algebra è una manipolazione di simboli che trova il suo fondamento nella correttezza formale, in algebra il significato delle operazioni e dei risultati dipende solo dai postulati assunti e non dalle interpretazione dei simboli e quindi l'algebra può essere considerata come un primo passo verso la logica. In effetti la logica, come presentata da Boole e De Morgan, è una parte dell'algebra. Inoltre la logica del I ordine richiede la padronanza delle nozioni di relazione, funzione, operazione, che sono argomenti propri dell'algebra e di contro le teorie algebriche costituiscono ottimi esempi di teorie logiche del I ordine, come vedremo verso la fine del corso.

Utilizzeremo fin dall'inizio le notazioni della teoria degli insiemi, anche se come vedremo in seguito la teoria degli insiemi andrebbe presentata come teoria logica con il proprio sistema di assiomi e questioni nate dalla teoria "ingenua" degli insiemi (vedi paradosso di Russell) sono stati una spinta allo sviluppo della logica.

Logica proposizionale

Mattone costitutivo del linguaggio naturale è la *proposizione*, frase compiuta che è sempre vera o falsa.
Una proposizione può essere:

- *atomica*
- *composta*: cioè costruita a partire da proposizioni atomiche usando connettivi

I connettivi che consideriamo sono:
$\sim$ (not), $\wedge$ (and), $\vee$ (or), $\Rightarrow$ (implica), $\Leftrightarrow$ (se e solo se).
Per indicare l'ordine in cui i connettivi sono applicati si utilizzano parentesi aperte e chiuse.

Sintassi.
Per costruire le proposizioni usiamo quindi un linguaggio il cui
alfabeto è costituito da:

- lettere enunciative: A,B,…, (al più una infinità numerabile e quindi spesso indicate con A_i)
- connettivi: $\sim$, $\wedge$, $\vee$, $\Rightarrow$, $\Leftrightarrow$,
- simboli ausiliari: (,)

N.B. Spesso si aggiungono alle lettere enunciative altri due simboli $\bot$ e $\top$.

Tra le possibili sequenze di simboli scegliamo quelle che corrispondono ad una buona struttura di proposizioni composte e che chiamiamo **formule ben formate** (f.b.f).
Le f.b.f. sono definite in modo ricorsivo così:

- ogni lettera enunciativa è una f.b.f.,
- se $\mathcal{A}$ è una f.b.f. anche $(\sim\mathcal{A})$ è una f.b.f.,
- se $\mathcal{A}$ e $\mathcal{B}$ sono f.b.f. anche $(\mathcal{A} \wedge \mathcal{B})$, $(\mathcal{A} \vee \mathcal{B})$, $(\mathcal{A} \Rightarrow \mathcal{B})$, $(\mathcal{A} \Leftrightarrow \mathcal{B})$ sono f.b.f,
- niente altro è una f.b.f.

N.B. Le lettere stampate sono usate per denotare le lettere enunciative, mentre quelle corsive denotano una qualunque f.b.f.

Esempio.
((~(A ∧B))⇔(A⇒(B∨A))) è una f.b.f.
Ma contiene troppe parentesi!

Per evitare di dover scrivere tutte queste parentesi si fissa una **precedenza nell'uso dei connettivi**: se non altrimenti indicato dalle parentesi si ha

- ~ precede ∧ che precede ∨ che precede ⇒ che precede ⇔
- connettivi uguali si intendono associati a sinistra

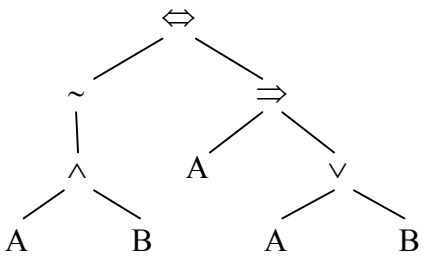
quindi la formula precedente può essere scritta come
~(A ∧B) ⇔ A⇒B∨A.

Questa formula è costruita mettendo insieme **sottoformule** che sono:
~(A ∧B) ⇔ A⇒B∨A, ~(A ∧B), A⇒B∨A, A ∧B, A, B∨A, B.

In genere data una formula $\mathcal{A}$ le **sottottoformule di $\mathcal{A}$, Stfm($\mathcal{A}$),** sono così definite:

- se $\mathcal{A}$ è lettera enunciativa, Stfm($\mathcal{A}$)={$\mathcal{A}$},
- se $\mathcal{A}$ è ~ $\mathcal{B}$, Stfm($\mathcal{A}$)={$\mathcal{A}$}∪ Stfm($\mathcal{B}$),
- se $\mathcal{A}$ è $\mathcal{B}$∧$\mathcal{C}$, $\mathcal{B}$∨$\mathcal{C}$, $\mathcal{B}$⇒$\mathcal{C}$, $\mathcal{B}$⇔$\mathcal{C}$, Stfm($\mathcal{A}$)={$\mathcal{A}$}∪ Stfm($\mathcal{B}$)∪Stfm($\mathcal{C}$).

L'**albero di struttura di una formula** rappresenta il modo in cui la formula è costruita ed evidenzia le sottoformule della formula stessa**.** Illustriamo quanto sopra attraverso un esempio.
La formula ~(A ∧B) ⇔ A⇒B∨A ha come albero di struttura :



Questo albero ha come radice l'ultimo connettivo usato (**connettivo principale**), come foglie le lettere enunciative, come nodi interni i connettivi.
Ogni nodo può essere visto come la radice di un sottoalbero massimale che è l'albero di struttura di una sottoformula e viceversa ogni sottoformula (che non sia una lettera enunciativa) ha come albero di struttura un sottoalbero massimale che ha come radice un nodo etichettato dal connettivo principale della sottoformula considerata.

Per come le f.b.f. sono definite appare chiaro che uno strumento importante per dimostrare che le f.b.f. godono di una data proprietà è l'induzione **sulla complessità di una proposizione**:

- Se una proprietà vale per tutte le lettere enunciative e se, supposta vera per $\mathcal{A}$ e $\mathcal{B}$, vale per ~ $\mathcal{A}$, $\mathcal{A}$∧$\mathcal{B}$, $\mathcal{A}$∨$\mathcal{B}$, $\mathcal{A}$⇒$\mathcal{B}$, $\mathcal{A}$⇔$\mathcal{B}$, allora la proprietà vale per tutte le formule.

Semantica

ovvero come stabilire che valore una f.b.f. a partire dai valori delle lettere enunciative che vi compaiono.

Si dice **interpretazione** una funzione v: f.b.f.$\rightarrow\{0,1\}$ che soddisfi le seguenti proprietà:

- $v(\perp)=0$
- $v(\mathsf{T})=1$
- $v(\sim \mathcal{A})=1-v(\mathcal{A})$
- $v(\mathcal{A}\wedge\mathcal{B})= \min(v(\mathcal{A}),v(\mathcal{B}))$
- $v(\mathcal{A}\vee\mathcal{B})= \max(v(\mathcal{A}),v(\mathcal{B}))$
- $v(\mathcal{A}\Rightarrow\mathcal{B})= \max(1-v(\mathcal{A}),v(\mathcal{B}))$
- $v(\mathcal{A}\Leftrightarrow\mathcal{B})= \min(\max(1-v(\mathcal{A}),v(\mathcal{B})), \max(v(\mathcal{A}),1-v(\mathcal{B})))$.

E' facile rendersi conto che fissare una interpretazione corrisponde a:

- attribuire arbitrariamente un valore di verità (1 o 0) a tutte le lettere enunciative (eccetto ai due simboli speciali $\perp$ e T che devono avere rispettivamente valori 0 e 1)
- passare dal valore di verità di $\mathcal{A}$ e $\mathcal{B}$ a quello di $\sim\mathcal{A}$, $\mathcal{A}\wedge\mathcal{B}$, $\mathcal{A}\vee\mathcal{B}$, $\mathcal{A}\Rightarrow\mathcal{B}$, $\mathcal{A}\Leftrightarrow\mathcal{B}$, utilizzando le tavole di verità dei connettivi.

I connettivi hanno le seguenti tavole di verità:

A	~A
0	1
1	0

A	B	A∧B
0	0	0
0	1	0
1	0	0
1	1	1

A	B	A∨B
0	0	0
0	1	1
1	0	1
1	1	1

A	B	A⇒B
0	0	1
0	1	1
1	0	0
1	1	1

A	B	A⇔B
0	0	1
0	1	0
1	0	0
1	1	1

Una f.b.f. $\mathcal{A}$ si dice **soddisfacibile** se esiste almeno una interpretazione v tale che $v(\mathcal{A})=1$. L'interpretazione v si dice in tal caso **modello** di $\mathcal{A}$.

Una fbf $\mathcal{A}$ per cui ogni interpretazione è un modello si dice **tautologia** e si scrive $\models\mathcal{A}$.

Una fbf $\mathcal{A}$ che non ammette modelli si dice **insoddisfacibile**.

N.B. $\mathcal{A}$ è una tautologia se e solo se $\sim\mathcal{A}$ è insoddisfacibile.

A questo punto se vogliamo ottenere tutte le possibili interpretazioni della formula del nostro esempio: $\sim(A \wedge B) \Leftrightarrow (A\Rightarrow(B\vee A))$ possiamo costruire la seguente tavola di verità, tramite la quale si arriva a calcolare il valore della formula passando attraverso le valutazioni delle sue sottoformule dalle più semplici alle più complesse:

A	B	A∧B	B∨A	~(A ∧B)	A⇒(B∨A)	~(A ∧B) ⇔ (A⇒(B∨A))
0	0	0	0	1	1	1
0	1	0	1	1	1	1
1	0	0	1	1	1	1
1	1	1	1	0	1	0

Più brevemente possiamo calcolare la tavola di verità della nostra formula, senza riscrivere tutte le sue sottoformule, ma mettendo i valori di verità di ogni sottoformula sotto il suo connettivo principale (cioè sotto l'ultimo connettivo usato per costruire la sottoformula):

~ (A	∧ B)	⇔	(A ⇒ (B ∨ A))
1 0	0 0	**1**	0 1 0 0 0
1 0	0 1	**1**	0 1 1 1 0
1 1	0 0	**1**	1 1 0 1 1
0 1	1 1	**0**	1 1 1 1 1

Le righe della tavola di verità sono tutte le possibili interpretazioni della formula e le righe che restituiscono il valore 1 (nella colonna sotto il connettivo principale della formula che è sopra evidenziata in grassetto) sono i modelli della formula.
Sono allora modelli per la nostra formula le valutazioni :
v_1 per cui $v_1(A)= v_1(B)=0$,
v_2 per cui $v_2(A)=0$, $v_2(B)=1$,
v_3 per cui $v_3(A)=1$, $v_3(B)=0$.

Decidere se una f.b.f. è soddisfacibile richiede un procedimento semplice, ma costoso dal punto di vista della complessità (tale problema è infatti NP-completo).

I concetti di modello, soddisfacibilità e insoddisfacibilità si possono estendere ad un insieme Γ di f.b.f.:

- un modello per Γ è una interpretazione che sia modello per ogni f.b.f. di Γ
- Γ è soddisfacibile se ammette un modello
- Γ è insoddisfacibile se nessuna interpretazione è un modello per Γ.

Una f.b.f. $\mathcal{A}$ è **conseguenza semantica** di un insieme Γ di f.b.f. , e si scrive $\Gamma \models \mathcal{A}$, se ogni modello di Γ è un modello per $\mathcal{A}$.
In particolare $\mathcal{A}$ è *conseguenza semantica* di $\mathcal{B}$ se ogni modello di $\mathcal{B}$ è modello di A .

Si ottiene subito il seguente

Teorema di deduzione semantica:
$\mathcal{A}$ è *conseguenza semantica* di $\mathcal{B}$ se e solo se $\mathcal{B}\Rightarrow\mathcal{A}$ è una tautologia.

che può essere scritto in una forma più generale

Teorema di deduzione semantica:
$\mathcal{A}$ è conseguenza semantica di $\Gamma\cup\{ \mathcal{B} \}$ se e solo se $\mathcal{B}\Rightarrow\mathcal{A}$ è conseguenza semantica di Γ.
Dim:
Ip. $\Gamma\cup\{\mathcal{B}\} \models \mathcal{A}$ Ts. $\Gamma \models \mathcal{B}\Rightarrow\mathcal{A}$

Sia v un modello per Γ. Distinguiamo due casi.

1. Se v è un modello per $\mathcal{B}$, v è anche un modello per $\Gamma\cup\{\mathcal{B}\}$ e dall'ipotesi si ha $v(\mathcal{A})=1$, quindi $v(\mathcal{B}\Rightarrow\mathcal{A})=1$.
2. Se v non è un modello per $\mathcal{B}$ cioè se $v(\mathcal{B})=0$,si ha $v(\mathcal{B}\Rightarrow\mathcal{A}) = \max(1-v(\mathcal{B}),v(\mathcal{A})) = 1$.

In entrambi i casi quindi v è un modello per $\mathcal{B}\Rightarrow\mathcal{A}$.

Ip. $\Gamma \models \mathcal{B}\Rightarrow\mathcal{A}$ Ts. $\Gamma\cup\{\mathcal{B}\} \models \mathcal{A}$

Sia v un modello per $\Gamma\cup\{\mathcal{B}\}$, allora v è un modello per Γ e per $\mathcal{B}$. Essendo un modello per Γ dall'ipotesi si ha $v(\mathcal{B}\Rightarrow\mathcal{A})=1$ che assieme a $v(\mathcal{B})=1$ implica $v(\mathcal{A})=1$.

E' interessante il legame fra deduzione semantica ed insoddisfacibilità, dato dal seguente

Teorema:

$\mathcal{A}$ è conseguenza semantica di Γ se e solo se $\Gamma\cup\{\sim\mathcal{A}\}$ è insoddisfacibile.

Dim.

Ip . $\Gamma \models \mathcal{A}$ Ts. $\Gamma\cup\{\sim\mathcal{A}\}$ è insoddisfacibile

Sia v una qualunque interpretazione. Distinguiamo due casi.

1. Se v è un modello per Γ dall'ipotesi si ha $v(\mathcal{A})=1$ e quindi $v(\sim\mathcal{A}) = 0$, quindi v non è un modello per $\Gamma\cup\{\sim\mathcal{A}\}$.
2. Se v non è un modello per Γ, non può essere sicuramente un modello per un insieme di formule che lo contiene.

Ip. $\Gamma\cup\{\sim\mathcal{A}\}$ è insoddisfacibile Ts. $\Gamma \models \mathcal{A}$

Sia v un modello per Γ, allora non dovendo essere un modello per $\Gamma\cup\{\sim\mathcal{A}\}$ si ha $v(\sim\mathcal{A})=0$ e quindi $v(\mathcal{A})=1$, dunque ogni modello di Γ è modello per $\mathcal{A}$.

Notiamo che non abbiamo mai imposto limiti sulla cardinalità di Γ. A tal proposito se Γ è un insieme infinito è importante il

Teorema di compattezza:

Un insieme Γ di formule è soddisfacibile se e solo se ogni suo sottoinsieme finito è soddisfacibile

Una formula $\mathcal{A}$ è **semanticamente equivalente** a $\mathcal{B}$ (scriveremo $\mathcal{A}\equiv\mathcal{B}$) se tutti e soli i modelli di $\mathcal{A}$ sono modelli di $\mathcal{B}$, in altre parole se $\mathcal{A}$ è *conseguenza semantica* di $\mathcal{B}$ e $\mathcal{B}$ è *conseguenza semantica* di $\mathcal{A}$.

$\mathcal{A}$ è semanticamente equivalente a $\mathcal{B}$ se e solo se $\mathcal{A}\Leftrightarrow\mathcal{B}$ è una tautologia.

Le formule non semanticamente equivalenti costruite utilizzando n lettere enunciative sono al più 2^{2^n}, tante sono infatti le possibili tavole di verità distinte poiché ci sono 2^n possibili assegnamenti di valori alle lettere enunciative e per ogni assegnamento di valori alle lettere enunciative ci sono due possibili valori di verità per una f.b.f.

Ad ogni tavola di verità corrisponde sempre una f.b.f. che la ammette come tavola di verità?
La risposta è affermativa ed il metodo per costruire la formula è il seguente:
ad ogni riga della tavola si associa un termine costruito facendo l'and di ogni lettera enunciativa o della sua negazione, a seconda che le lettere assumano in quella riga il valore 1 o 0 , poiché i termini così costruiti hanno la caratteristica di valere 1 solo in corrispondenza dell'assegnamento di valori di verità fissato da quella riga, facendo l'or dei termini che corrispondono alle righe in cui la formula assume il valore 1, si costruisce una f.b.f. che ha esattamente la tavola di verità assegnata. La formula così costruita ha una forma speciale detta forma normale disgiuntiva.

Esempio
Sia data la tavola

A	B	C	f(A,B,C)
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

Una formula (in forma normale disgiuntiva) che ammette la tavola data come tavola di verità è $(\sim A \wedge B \wedge \sim C) \vee (\sim A \wedge B \wedge C) \vee (A \wedge B \wedge \sim C)$

Possiamo quindi osservare che una qualsiasi f.b.f. ammette una formula equivalente che usa solo i tre connettivi $\sim$, $\wedge$, $\vee$.

Per trasformare una formula in modo da costruire una f.b.f. equivalente alla data e che sia "più semplice" perché utilizza o un numero minore di connettivi, o un numero minore di tipi di connettivi, si utilizzano le seguenti osservazioni:

- se in una formula $\mathcal{A}$ si sostituisce una sottoformula $\mathcal{B}$ con una formula $\mathcal{B}'$ equivalente a $\mathcal{B}$, si ottiene una formula $\mathcal{A}'$ equivalente ad $\mathcal{A}$;

- se in una tautologia $\mathcal{A}$ si sostituisce ogni occorrenza di una stessa lettera enunciativa A con una stessa formula $\mathcal{B}$, si ottiene ancora una tautologia;

e le seguenti equivalenze fondamentali:

$\sim(\sim A) \equiv A$

$A \wedge A \equiv A$	$A \vee A \equiv A$
$A \wedge B \equiv B \wedge A$	$A \vee B \equiv B \vee A$
$(A \wedge B) \wedge C \equiv A \wedge (B \wedge C)$	$(A \vee B) \vee C \equiv A \vee (B \vee C)$
$A \wedge (A \vee B) \equiv A$	$A \vee (A \wedge B) \equiv A$
$A \wedge (B \vee C) \equiv (A \wedge B) \vee (A \wedge C)$	$A \vee (B \wedge C) \equiv (A \vee B) \wedge (A \vee C)$
$\sim(A \wedge B) \equiv \sim A \vee \sim B$	$\sim(A \vee B) \equiv \sim A \wedge \sim B$
$A \Rightarrow B \equiv \sim A \vee B$	$A \Rightarrow B \equiv \sim(A \wedge \sim B)$
$B \equiv (\sim A \wedge A) \vee B$	$B \equiv (\sim A \vee A) \wedge B$

(le ultime due formule possono essere sostituite rispettivamente con $\bot \equiv \sim A \wedge A$ e $\bot \vee B \equiv B$; $\mathsf{T} \equiv \sim A \vee A$ e $\mathsf{T} \wedge B \equiv B$, che fanno uso dei due simboli speciali $\bot$ e T)

Usando la tavola delle equivalenze si può vedere che ogni f.b.f. ammette una f.b.f. equivalente che usa solo uno qualunque degli insiemi di connettivi $\{\sim,\wedge\}\{\sim,\vee\},\{\sim,\Rightarrow\}$, detti **insiemi adeguati (o funzionalmente completi) di connettivi**.

La formula $(\sim A \wedge B \wedge \sim C) \vee (\sim A \wedge B \wedge C) \vee (A \wedge B \wedge \sim C)$ del precedente esempio è equivalente a $((\sim A \wedge B) \wedge (\sim C \vee C)) \vee (A \wedge B \wedge \sim C)$ e quindi a $(\sim A \wedge B) \vee (A \wedge B \wedge \sim C)$, che a sua volta è equivalente a $B \wedge (\sim A \vee (A \wedge \sim C))$ e quindi a $B \wedge ((\sim A \vee A) \wedge (\sim A \vee \sim C))$ cioè a $B \wedge (\sim A \vee \sim C)$.
Se volessimo eliminare il connettivo $\vee$, si vede facilmente che la nostra formula è equivalente a $B \wedge \sim (A \wedge C)$; analogamente se volessimo eliminare il connettivo $\wedge$, potremmo scrivere la formula equivalente alla data $\sim(\sim B \vee \sim (\sim A \vee \sim C))$.
Infine se la volessimo scrivere in una forma equivalente usando solo i connettivi $\sim, \Rightarrow$ avremmo ad esempio $\sim((A \Rightarrow \sim C) \Rightarrow \sim B)$.

N.B. Gli insiemi adeguati di connettivi non possono essere ulteriormente ridotti, a meno di non introdurre i nuovi connettivi $|$ (nor) e $\downarrow$ (nand), che hanno le seguenti tavole di verità

A	B	A | B
0	0	1
0	1	0
1	0	0
1	1	0

A	B	A↓B
0	0	1
0	1	1
1	0	1
1	1	0

Tenendo conto che si hanno le seguenti equivalenze,
$\sim A$ equivalente ad $A|A$ ed ad $A \downarrow A$
$A \wedge B$ equivalente ad $(A \downarrow B) \downarrow (A \downarrow B)$
$A \vee B$ equivalente ad $(A|B)|(A|B)$
ogni f.b.f ammette una formula equivalente che usa solo i connettivi nor o nand.

A questo punto potrebbe risultare utile (ad esempio in ambito dell'intelligenza artificiale) un sistema puramente sintattico di manipolazione di f.b.f., che permetta di ricavare tutte e sole le tautologie (sia cioè completo e corretto) e, partendo da un insieme arbitrario di formule, tutte e sole le formule che ne sono conseguenza semantica. Per far questo dobbiamo introdurre la nozione di

TEORIA FORMALE (o Sistema deduttivo)

Una teoria formale è definita quando sono fissati:
- un insieme di simboli (**alfabeto**),
- un insieme di stringhe privilegiate di simboli (**f.b.f.**),
- un insieme privilegiato di f.b.f. (**assiomi** o **base della conoscenza**) e
- un insieme di regole di riscrittura (o di **inferenza**) che in presenza di un certo insieme di f.b.f . permetta di scriverne in modo algoritmico altre (inferite o dedotte dalle precedenti).

Data una teoria formale H (cioè specificati tutti gli insiemi precedentemente elencati), chiamiamo **dimostrazione nella teoria formale** H una sequenza finita di f.b.f. che siano o assiomi o formule dedotte dalle precedenti tramite le regole di inferenza, diciamo **teorema della teoria** una f.b.f. $\mathcal{A}$ (e scriviamo $|-_H \mathcal{A}$) che sia l'ultima formula di una dimostrazione.

Dato un insieme Γ di f.b.f. diciamo che una formula $\mathcal{A}$ è (sintatticamente) **deducibile** in H da Γ (e scriviamo $\Gamma|-_H \mathcal{A}$) se esiste una sequenza finita di f.b.f. che siano o assiomi o formule di Γ o formule dedotte dalle precedenti tramite le regole di inferenza, la cui ultima formula sia $\mathcal{A}$.
Un teorema di H è dunque una formula deducibile da un insieme vuoto di f.b.f.

Osserviamo che se $\Gamma|\text{-}_H \mathcal{A}$ allora

- esiste un insieme finito $\Gamma' \subseteq \Gamma$ tale che $\Gamma'|\text{-}_H \mathcal{A}$
- per ogni insieme di f.b.f. Δ tale che $\Delta \supseteq \Gamma$, si ha $\Delta|\text{-}_H \mathcal{A}$.

Vogliamo a questo punto definire una teoria sostanzialmente basata sul linguaggio che abbiamo introdotto all'inizio, che chiamiamo teoria L, che permetta di ottenere come teoremi tutte e sole le tautologie e permetta di dedurre da un insieme Γ di formule tutte e sole le conseguenze semantiche di Γ.

Simboli di L:

- lettere enunciative: A,B,…,
- connettivi: ~, $\Rightarrow$,
- parentesi: (,)

Formule ben formate (f.b.f.) di L:

- lettere enunciative,
- se $\mathcal{A}$ è una f.b.f. anche $(\sim\mathcal{A})$ è f.b.f.,
- se $\mathcal{A}$ e $\mathcal{B}$ sono f.b.f. anche $(\mathcal{A}\Rightarrow\mathcal{B})$ è una f.b.f.
- niente altro è una f.b.f.

(In realtà si accettano tra le f.b.f. formule del tipo $(\mathcal{A}\wedge\mathcal{B})$, $(\mathcal{A}\vee\mathcal{B})$, $(\mathcal{A}\Leftrightarrow\mathcal{B})$, ma tali formule vengono pensate come abbreviazioni di una formula ad esse equivalente che usi solo i connettivi ~, $\Rightarrow$).

N.B. Al solito, se non diversamente indicato dalle parentesi, ~ precede $\Rightarrow$.

Assiomi di L:

A1. $\mathcal{A}\Rightarrow(\mathcal{B}\Rightarrow\mathcal{A})$

A2. $(\mathcal{A}\Rightarrow(\mathcal{B}\Rightarrow\mathcal{C}))\Rightarrow((\mathcal{A}\Rightarrow\mathcal{B})\Rightarrow(\mathcal{A}\Rightarrow\mathcal{C}))$

A3. $(\sim\mathcal{A}\Rightarrow\sim\mathcal{B})\Rightarrow((\sim\mathcal{A}\Rightarrow\mathcal{B})\Rightarrow\mathcal{A})$

N.B. A1, A2, A3 non sono tre formule ma tre schemi di formule perché al loro interno le sottoformule $\mathcal{A}$, $\mathcal{B}$, $\mathcal{C}$ sono qualsiasi.

Regola di inferenza di L:

Modus Ponens (MP). Dalle due formule $\mathcal{A}$ e $\mathcal{A}\Rightarrow\mathcal{B}$ si riscrive $\mathcal{B}$.

<u>Esempio</u>

$|\text{-}_L \mathcal{A}\Rightarrow\mathcal{A}$

Per dimostrarlo dobbiamo trovare una dimostrazione in L che finisca con la formula $\mathcal{A}\Rightarrow\mathcal{A}$ e tale che le formule della sequenza o siano assiomi o siano ricavate da formule precedenti per MP.

1. $\mathcal{A}\Rightarrow(\mathcal{A}\Rightarrow\mathcal{A})$ (è lo schema di assiomi A1 dove $\mathcal{B}$ è stato sostituito con $\mathcal{A}$)
2. $\mathcal{A}\Rightarrow((\mathcal{A}\Rightarrow\mathcal{A})\Rightarrow\mathcal{A})$ (è lo schema di assiomi A1 dove $\mathcal{B}$ è stato sostituito con $\mathcal{A}\Rightarrow\mathcal{A}$)
3. $(\mathcal{A}\Rightarrow((\mathcal{A}\Rightarrow\mathcal{A})\Rightarrow\mathcal{A}))\Rightarrow((\mathcal{A}\Rightarrow(\mathcal{A}\Rightarrow\mathcal{A}))\Rightarrow(\mathcal{A}\Rightarrow\mathcal{A}))$
   (è lo schema di assiomi A2 dove
   $\mathcal{B}$ è stato sostituito con $\mathcal{A}\Rightarrow\mathcal{A}$ e $\mathcal{C}$ con $\mathcal{A}$)
4. $(\mathcal{A}\Rightarrow(\mathcal{A}\Rightarrow\mathcal{A}))\Rightarrow(\mathcal{A}\Rightarrow\mathcal{A})$ (applicando MP fra le formule 2 e 3)
5. $\mathcal{A}\Rightarrow\mathcal{A}$ (applicando MP fra le formule 1 e la 4)

La teoria formale L che abbiamo così presentato ha tre importanti caratteristiche:

- è *corretta*, cioè tutti i suoi teoremi sono tautologie,
- è *completa*, cioè tutte le tautologie sono teoremi di L,
- è *decidibile*, cioè esiste un algoritmo (la tavola di verità) che, con un numero finito (di cui è noto il limite superiore) di passi, permette di decidere se una data formula è o non è un teorema della teoria.

Le prime due affermazioni vengono di solito chiamate *(meta)teoremi di correttezza e completezza* il prefisso meta indica che sono teoremi sulla teoria che sono enunciati e non sono dimostrati utilizzando il linguaggio della teoria stessa.

Accenniamo solo alla *dimostrazione del teorema di correttezza*:
E' immediato verificare che gli schemi di assiomi della teoria sono tutti tautologie, inoltre il M.P. fa passare da tautologie a tautologie, pertanto possiamo dimostrare che ogni teorema di L è una tautologia procedendo per induzione sul numero *n* di formule della dimostrazione.
Se n=1, la dimostrazione consiste di una sola formula (il teorema) che può essere solo un assioma e quindi è una tautologia.
Per ipotesi di induzione supponiamo che ogni formula dimostrata con un numero di passi $m<n$ sia una tautologia. Sia $\mathcal{A}$ un teorema dimostrato con n passi. Se $\mathcal{A}$ è scritta, come n-esimo passo della dimostrazione, in quanto assioma è ovviamente una tautologia; se invece è scritta perché si è utilizzato il MP su due formule precedenti, ognuna di queste, essendo stata dimostrata con un numero di passi inferiore ad n, è una tautologia per ipotesi di induzione ed $\mathcal{A}$ è una tautologia perché il M.P. fa passare da tautologie a tautologie.

I teoremi di correttezza e completezza ammettono anche una formulazione più forte che è la seguente:
Teorema di correttezza e completezza forte:
Sia Γ un insieme di f.b.f., $\Gamma \models \mathcal{A}$ se e solo se $\Gamma \vdash_L \mathcal{A}$.

Questo teorema può essere facilmente ricavato dalla versione debole dei teoremi di completezza e correttezza, utilizzando il teorema di deduzione semantica e il

Teorema di deduzione (sintattica):
Sia $\Gamma = \Delta \cup \{\mathcal{B}\}$ un insieme di f.b.f. $\Gamma \vdash_L \mathcal{A}$ se e solo se $\Delta \vdash_L \mathcal{B} \Rightarrow \mathcal{A}$.

Il teorema di deduzione sintattica è uno strumento molto utile nel cercare di stabilire se una formula è conseguenza sintattica di altre.

Esempio:
Provare che $\mathcal{A} \Rightarrow \mathcal{B} \vdash_L (\mathcal{B} \Rightarrow \mathcal{C}) \Rightarrow (\mathcal{A} \Rightarrow \mathcal{C})$.
Per il teorema di deduzione sintattica $\mathcal{A} \Rightarrow \mathcal{B} \vdash_L (\mathcal{B} \Rightarrow \mathcal{C}) \Rightarrow (\mathcal{A} \Rightarrow \mathcal{C})$ se e solo se $\mathcal{A} \Rightarrow \mathcal{B}, \mathcal{B} \Rightarrow \mathcal{C} \vdash_L \mathcal{A} \Rightarrow \mathcal{C}$; ma ancora per il teorema di deduzione sintattica $\mathcal{A} \Rightarrow \mathcal{B}, \mathcal{B} \Rightarrow \mathcal{C} \vdash_L \mathcal{A} \Rightarrow \mathcal{C}$ se e solo se $\mathcal{A} \Rightarrow \mathcal{B}, \mathcal{B} \Rightarrow \mathcal{C}, \mathcal{A} \vdash_L \mathcal{C}$.
Quest'ultima deduzione risulta molto semplice, infatti possiamo scrivere

1. $\mathcal{A}$ perché è una premessa,
2. $\mathcal{A} \Rightarrow \mathcal{B}$ perché è una premessa,
3. $\mathcal{B}$ perché è ottenuta per MP da 1 e 2,

4. $\mathcal{B}\Rightarrow\mathcal{C}$ perché è una premessa,
5. $\mathcal{C}$ perché è ottenuta per MP da 3 e 4.

Diamo a questo punto la
dimostrazione del teorema di deduzione sintattica
Ip: $\Delta\cup\{\mathcal{B}\}|\text{-}_L\,\mathcal{A}$ Ts: $\Delta|\text{-}_L\,\mathcal{B}\Rightarrow\mathcal{A}$
La dimostrazione procede per induzione sul numero *n* di formule che costituiscono la sequenza di deduzione di $\mathcal{A}$ da $\Delta\cup\{\mathcal{B}\}$.
Caso base *n*=1. In tal caso $\mathcal{A}$ è o un assioma o una formula di $\Delta\cup\{\mathcal{B}\}$.
Se $\mathcal{A}$ è un assioma o una formula di Δ allora possiamo costruire la sequenza di formule:

1. $\mathcal{A}$ (assioma o formula di Δ)
2. $\mathcal{A}\Rightarrow(\mathcal{B}\Rightarrow\mathcal{A})$ (assioma A1)
3. $\mathcal{B}\Rightarrow\mathcal{A}$ (M.P. da 1 e 2)

che è una deduzione di $\mathcal{B}\Rightarrow\mathcal{A}$ da Δ.
Se invece $\mathcal{A}$ è la formula $\mathcal{B}$, sappiamo già che $\mathcal{A}\Rightarrow\mathcal{A}$ è un teorema e, come tale, a maggior ragione è deducibile da Δ.
Ipotesi di induzione: il teorema vale per ogni formula deducibile da $\Delta\cup\{\mathcal{B}\}$ con una sequenza di formule di lunghezza inferiore ad *n*
Supponiamo ora che la deduzione di $\mathcal{A}$ da $\Delta\cup\{\mathcal{B}\}$ richieda *n* formule.
$\mathcal{A}$ sarà la *n*-esima formula della deduzione e sarà scritta nella sequenza perché è un assioma o una formula di $\Delta\cup\{\mathcal{B}\}$ o è ottenuta per M.P. da due formule precedenti .
Nei primi due casi si procede come nel caso base (notate tra l'altro che sarebbe stato del tutto inutile scrivere $\mathcal{A}$ come *n*-esima formula, si poteva scrivere subito come prima formula), supponiamo allora di avere nella nostra sequenza una formula $\mathcal{C}$ al posto *h*-esimo della sequenza con $h<n$ e una formula $\mathcal{C}\Rightarrow\mathcal{A}$ al posto *k*-esimo della sequenza con $k<n$.
Per ipotesi di induzione possiamo dedurre da Δ sia $\mathcal{B}\Rightarrow\mathcal{C}$ sia $\mathcal{B}\Rightarrow(\mathcal{C}\Rightarrow\mathcal{A})$, a questo punto consideriamo le sequenze di deduzione delle due formule da Δ

…
…
$\mathcal{B}\Rightarrow\mathcal{C}$
...
...
$\mathcal{B}\Rightarrow(\mathcal{C}\Rightarrow\mathcal{A})$

e ad esse aggiungiamo

$(\mathcal{B}\Rightarrow(\mathcal{C}\Rightarrow\mathcal{A}))\Rightarrow((\mathcal{B}\Rightarrow\mathcal{C})\Rightarrow(\mathcal{B}\Rightarrow\mathcal{A}))$	(assioma A2)
$(\mathcal{B}\Rightarrow\mathcal{C})\Rightarrow(\mathcal{B}\Rightarrow\mathcal{A})$	(M.P. fra le ultime due)
$\mathcal{B}\Rightarrow\mathcal{A}$	(M.P. fra l'ultima e $\mathcal{B}\Rightarrow\mathcal{C}$)

La sequenza così ottenuta è fatta tutta di assiomi o di formule di Δ o di formule ottenute per M.P. da due formule che le precedono e dunque è una deduzione di $\mathcal{B}\Rightarrow\mathcal{A}$ da Δ.
Ip: $\Delta|\text{-}_L\,\mathcal{B}\Rightarrow\mathcal{A}$ Ts: $\Delta\cup\{\mathcal{B}\}|\text{-}_L\,\mathcal{A}$
Basta banalmente scrivere $\mathcal{B}$, poi la deduzione di $\mathcal{B}\Rightarrow\mathcal{A}$ da Δ ed infine usare il M.P.

Ci si potrebbe chiedere a questo punto se tutti e tre gli schemi di assiomi di L siano necessari o se uno di essi potrebbe essere ricavato come teorema dai due restanti. La risposta è negativa, si può infatti dimostrare che i tre schemi sono indipendenti.
Per dimostrare l'indipendenza di A1 da A2 e A3 (e quella di A2 da A1 e A3) si utilizzano tavole di "verità" a 3 valori per i connettivi ~ e $\Rightarrow$. Tali tavole sono fatte in modo che M.P. faccia passare da formule il cui valore è 0 a formule in cui valore è 0 e che gli schemi A2 e A3 (A1 e A3) valgano 0, mentre l'assioma A1 (A2) può assumere valori diversi. Dunque A1 (A2) non può essere ricavato come teorema dai restanti assiomi, altrimenti dovrebbe valere sempre 0.
L'indipendenza di A3 da A1 e A2 si dimostra con tecnica diversa: si utilizza un operatore *h* che cancella da ogni formula il connettivo ~ e lascia fissi tutti gli altri simboli del linguaggio, *h* porta da f.b.f. a f.b.f. ed in particolare porta gli schemi A1 e A2 in formule che sono ancora istanza di A1 ed A2, mentre porta A3 in una formula del tipo $(h(\mathcal{A}) \Rightarrow h(\mathcal{B})) \Rightarrow ((h(\mathcal{A}) \Rightarrow h(\mathcal{B})) \Rightarrow h(\mathcal{A}))$, che non è un'istanza di A3 ed inoltre non è una tautologia. Se A3 si potesse dedurre come teorema da A1 e A2, operando con *h* su tutte le formule della sequenza dimostrativa, si troverebbero ancora istanze di A1 e A2 e formule ottenute da formule precedenti per M.P., allora si avrebbe una dimostrazione di *h*(A3) in L e quindi *h*(A3) dovrebbe essere una tautologia mentre, come abbiamo notato, non lo è.

Da ultimo vogliamo osservare che la teoria L è solo uno dei tanti sistemi formali che si possono introdurre per la logica proposizionale. Si hanno numerosi altri sistemi, basati su scelte diverse degli insiemi di connettivi, degli assiomi, alcuni di questi presentano un numero inferiore di assiomi, anche se a prezzo di una più complessa struttura sintattica delle formule scelte come assiomi.
La teoria L, come gli altri sistemi assiomatici, non è sicuramente un sistema formale che ben si presta alla dimostrazione automatica, perché richiede "fantasia" nella scelta delle istanze di assiomi da usare. Uno dei sistemi più diffusi nell'ambito della dimostrazione automatica grazie alla "meccanicità" d'uso è la **risoluzione**.

La risoluzione è alla base del PROLOG e verifica se una formula $\mathcal{A}$ sia una tautologia (e quindi un teorema di L) o se sia deducibile da un insieme di formule Γ, provando tramite tecniche di riscrittura, rispettivamente l'insoddisfacibilità di $\sim\mathcal{A}$, o di $\Gamma \cup \{\sim\mathcal{A}\}$.

Iniziamo ad introdurre un po' di terminologia:

- si dice **letterale** una lettera enunciativa o la negazione di una lettera enunciativa
- si dice **clausola** la disgiunzione (finita) di letterali;
- una clausola viene rappresentata come insieme di letterali; una clausola che non contenga letterali si dice **clausola vuota** e si indica con $\square$
- una f.b.f. si dice in **forma a clausole** se è scritta come congiunzione di clausole ed in tal caso sarà denotata come insieme di insiemi

Ovviamente ogni formula ammette una formula equivalente in forma a clausole.

Esempio.
Si scriva in forma a clausole la f.b.f. $((A \Rightarrow B) \wedge (A \Leftrightarrow C)) \vee \sim B$
Si ha $((A \Rightarrow B) \wedge (A \Leftrightarrow C)) \vee \sim B \equiv ((\sim A \vee B) \wedge (\sim A \vee C) \wedge (\sim C \vee A)) \vee \sim B \equiv ((\sim A \vee B \vee \sim B) \wedge (\sim A \vee C \vee \sim B) \wedge (\sim C \vee A \vee \sim B)$
La forma a clausole della formula iniziale viene scritta come insieme di clausole, dove ogni clausola è indicata a sua volta come insieme di letterali:

$\{\{\sim A, B, \sim B\}, \{\sim A, C, \sim B\}, \{\sim C, A, \sim B\}\}$.

Date le clausole C_1, C_2 ed R, si dice che R è una **risolvente** di C_1 e C_2 se esiste un letterale $L \in C_1$ tale che $\sim L \in C_2$ ed $R=(C_1\setminus\{L\})\cup(C_2\setminus\{\sim L\})$ dove se L è una lettera enunciativa A il simbolo $\sim L$ sta per $\sim A$, se invece L è la negazione di A il simbolo $\sim L$ sta per A.
E' immediato verificare che se R è una risolvente di C_1 e C_2 si ha $C_1, C_2 \models R$.

Sia Γ un insieme di clausole, una **derivazione per risoluzione** della clausola C da Γ ($\Gamma \vdash_R C$) è una sequenza di clausole di cui l'ultima è C e che o stanno in Γ o sono ottenute come risolvente da clausole precedenti (si tratta quindi di una deduzione da Γ in un sistema formale con una sola regola di inferenza che fa passare da due clausole ad una loro risolvente).

Diamo ora una condizione necessaria e sufficiente affinché un insieme di clausole e quindi ogni f.b.f. associata a quell'insieme di clausole sia insoddisfacibile.

Teorema
Un insieme di clausole Γ è insoddisfacibile se e solo se $\Gamma \vdash_R \square$

Dim .

Ip. $\Gamma \vdash_R \square$ Ts. Γ è insoddisfacibile
Se Γ ammettesse un modello v, v sarebbe modello per tutte le clausole derivate da Γ e quindi anche per la clausola vuota che invece è sempre insoddisfacibile. Quindi Γ non ammette alcun modello e pertanto è insoddisfacibile.

Ts. Γ è insoddisfacibile Ip. $\Gamma \vdash_R \square$

Se Γ è insoddisfacibile per il teorema di compattezza esiste un suo sottoinsieme finito Δ insoddisfacibile e quindi basta dimostrare che $\Delta \vdash_R \square$, che ovviamente implica $\Gamma \vdash_R \square$.

Procediamo per induzione sul numero n di lettere enunciative occorrenti (eventualmente negate) nelle clausole di Δ.

Caso base $n=0$, Δ contiene solo la clausola vuota, che quindi è una clausola di Δ dunque $\Delta \vdash_R \square$.

Ipotesi di induzione : da ogni insieme di clausole insoddisfacibile che contenga meno di n lettere enunciative si deriva per risoluzione la clausola vuota.

Passo induttivo. Supponiamo che in Δ occorrano n lettere enunciative. Sia A una di queste n lettere. Dividiamo Δ in 3 sottoinsiemi, l'insieme Δ_0 (eventualmente vuoto) delle clausole in cui occorrono sia A sia $\sim A$, l'insieme Δ_1 delle clausole in cui non compare $\sim A$, l'insieme Δ_2 delle clausole in cui non compare A. Cancelliamo il letterale A da tutte le clausole di Δ_1 in cui compare ed il letterale $\sim A$ da tutte le clausole di Δ_2 in cui compare, ottenendo così rispettivamente i due insiemi di clausole Δ' e Δ''. Tali insiemi Δ' e Δ'' sono insoddisfacibili, infatti, se v fosse un modello per Δ', ponendo $v(A)=0$, v sarebbe un modello per Δ e se invece w fosse un modello per Δ'', ponendo $w(A)=1$, w sarebbe un modello per Δ; inoltre Δ' e Δ'' contengono un numero di lettere enunciative inferiore ad n dunque per ipotesi di induzione $\Delta' \vdash_R \square$ e $\Delta'' \vdash_R \square$. Se ora ripristiniamo A in tutte le clausole di Δ' da cui era stato cancellato e ripercorriamo la derivazione otteniamo o ancora $\Delta_1 \vdash_R \square$ (perché nelle clausole usate non occorreva A) o $\Delta_1 \vdash_R \{A\}$. Analogamente se ripristiniamo $\sim A$ in tutte le clausole di Δ'' da cui era stato cancellato e ripercorriamo la derivazione otteniamo o ancora $\Delta_2 \vdash_R \square$ (perché nelle clausole usate non occorreva $\sim A$) o $\Delta_2 \vdash_R \{\sim A\}$. Se $\Delta_1 \vdash_R \square$ o $\Delta_2 \vdash_R \square$ allora ovviamente

$\Delta \vdash_R \square$. Se invece $\Delta_1 \vdash_R \{A\}$ e $\Delta_2 \vdash_R \{\sim A\}$, abbiamo, con una ulteriore risoluzione fra le clausole $\{A\}$ e $\{\sim A\}$, $\Delta \vdash_R \square$.

Da questo ricaviamo che una formula $\mathcal{A}$ è semanticamente deducibile da un insieme di f.b.f. Γ se e solo se $\Gamma^c \cup \{\sim\mathcal{A}\}^c \vdash_R \square$ (dove Γ^c e $\{\sim\mathcal{A}\}^c$ sono rispettivamente l'insieme delle clausole ottenute dalle formule di Γ e la forma a clausole di $\sim\mathcal{A}$).

In conclusione:

- la risoluzione agisce per refutazione e opera su f.b.f. in forma a clausole
- è un sistema corretto ed è completo per refutazione,

ma se $\Gamma \models \mathcal{A}$ non è detto che $\Gamma^c \vdash_R \mathcal{A}^c$, dove Γ^c l'insieme delle clausole ottenute dalle formule di Γ ed $\mathcal{A}^c$ è la forma a clausole di $\mathcal{A}$, basta pensare $\Gamma=\{\{A\}\}$ ed $\mathcal{A}$ come $A \vee B$.

Per verificare se una clausola (in particolare la clausola vuota) si può ottenere per risoluzione da un insieme Γ di clausole è utile introdurre la seguente definizione:
$Ris(\Gamma)=\Gamma \cup \{C_{ij} | C_{ij}$ è risolvente di $C_i, C_j \in \Gamma\}$, $Ris^n(\Gamma)=Ris(Ris^{n-1}(\Gamma))$,
$Ris^*(\Gamma)= \cup_{n>0} Ris^n(\Gamma)$.
Osserviamo allora che $\Gamma \vdash_R C$ se e solo se $C \in Ris^*(\Gamma)$. In conclusione abbiamo il seguente algoritmo per stabilire se $\Gamma \models \mathcal{A}$:

1) trasformare le formule di Γ in forma a clausole ottenendo un insieme Γ^c di clausole
2) trasformare $\sim\mathcal{A}$ in forma a clausole $(\sim\mathcal{A})^c$
3) $S:=\Gamma^c \cup \{(\sim\mathcal{A})c\}$
   ripetere:
   (a) $F:=S$
   (b) $S:=Ris(S)$
   finché $\square \in S$ o $S=F$
4) se $\square \in S$ allora $\Gamma \models \mathcal{A}$, altrimenti $\mathcal{A}$ non è conseguenza semantica di Γ.

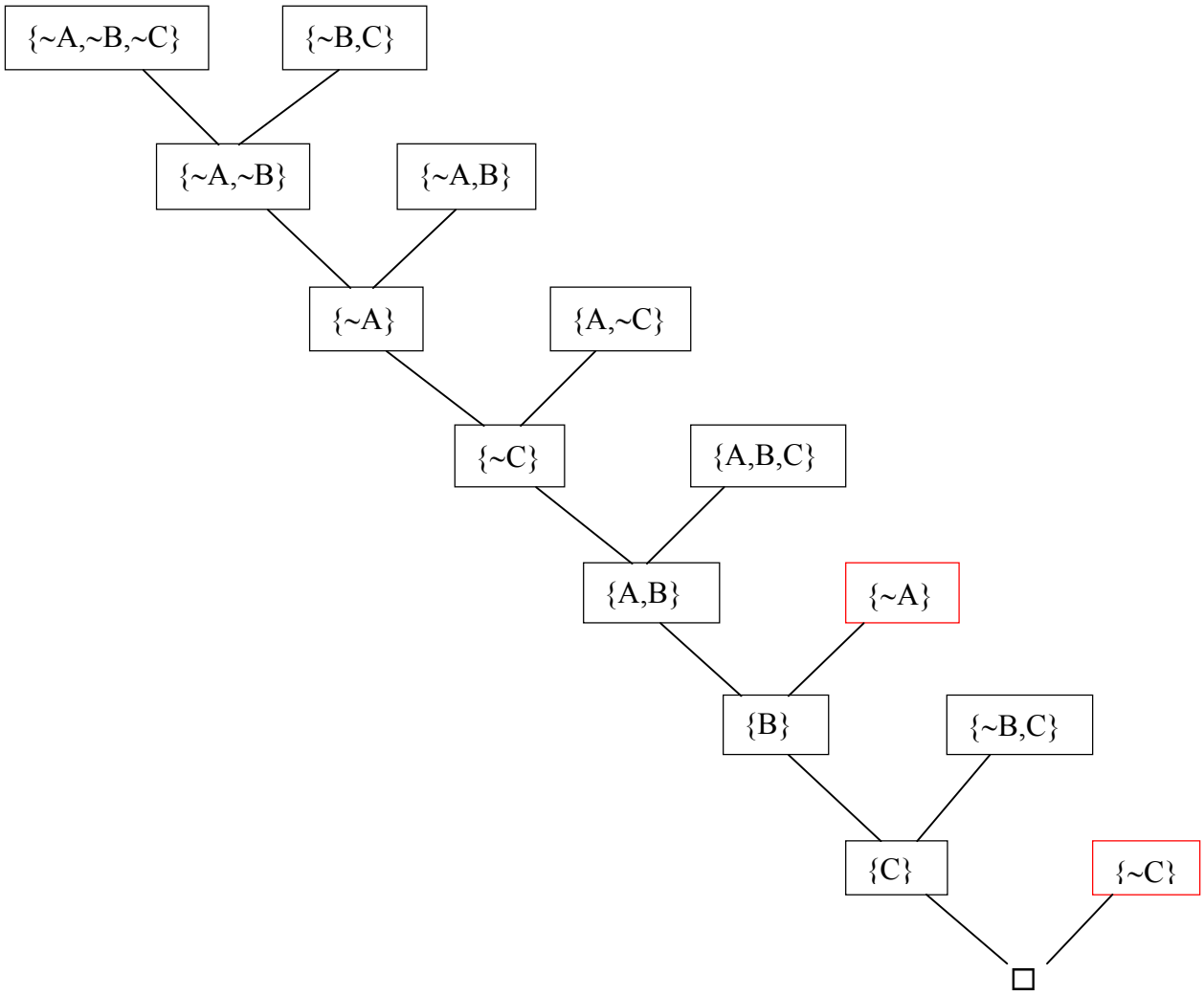
<u>Esempio</u>
Dire, applicando l'algoritmo, se $A \wedge B \wedge D$ è conseguenza semantica di
$(\sim B \vee C) \wedge \sim(A \wedge \sim B) \wedge (A \vee ((B \vee C) \wedge \sim C))$

Trasformiamo $(\sim B \vee C) \wedge \sim(A \wedge \sim B) \wedge (A \vee ((B \vee C) \wedge \sim C))$ in forma in clausole:
$(\sim B \vee C) \wedge \sim(A \wedge \sim B) \wedge (A \vee ((B \vee C) \wedge \sim C)) \equiv (\sim B \vee C) \wedge (\sim A \vee B) \wedge (A \vee B \vee C) \wedge (A \vee \sim C)$ da cui
$((\sim B \vee C) \wedge \sim(A \wedge B) \wedge (A \vee ((B \vee C) \wedge \sim C)))^c = \{\{\sim B, C\}, \{\sim A, B\}, \{A, B, C\}, \{A, \sim C\}\}$
trasformiamo $\sim(A \wedge B \wedge D)$ in forma a clausole e ottengo $\{\{\sim A, \sim B, \sim C\}\}$
$S:= \{\{\sim B, C\}, \{\sim A, B\}, \{A, B, C\}, \{A, \sim C\}, \{\sim A, \sim B, \sim C\}\}$, calcoliamo $Ris(S)$
$Ris(S) := \{\{\sim B, C\}, \{\sim A, B\}, \{A, B, C\}, \{A, \sim C\}, \{\sim A, \sim B, \sim C\}, \{\sim A, C\}, \{A, C\}, \{A, \sim B\}, \{\sim A, \sim B\}, \{B, C\}, \{B, \sim C\}, \{\sim A, \sim C\}, \{A, B\}, \{B, \sim B, C, \sim C\}, \{A, \sim A, C, \sim C\}, \{A, \sim A, B, \sim B\}, \{\sim B, \sim C\}\}$.
S è contenuto in $Ris(S)$ e $\square \notin S$, quindi calcoliamo
$Ris^2(S) := \{\{\sim B, C\}, \{\sim A, B\}, \{A, B, C\}, \{A, \sim C\}, \{\sim A, \sim B, \sim C\}, \{\sim A, C\}, \{A, C\}, \{A, \sim B\}, \{\sim A, \sim B\}, \{B, C\}, \{B, \sim C\}, \{\sim A, \sim C\}, \{A, B\}, \{B, \sim B, C, \sim C\}, \{A, \sim A, C, \sim C\}, \{A, \sim A, B, \sim B\}, \{\sim B, \sim C\}, \{C\}, \{C, \sim C\}, \{B, \sim B\}, \ldots, \{\sim A\}, \ldots, \{B\}, \ldots, \{A\}, \ldots\}$.
A questo punto sappiamo che al più $Ris^3(S)$ contiene $\square$ perché è ad esempio la risolvente di $\{\sim A\}$ e $\{A\}$.
Da questo deduciamo che $(\sim B \vee C) \wedge \sim(A \wedge \sim B) \wedge (A \vee ((B \vee C) \wedge \sim C)) \models A \wedge B \wedge D$.

Si osservi che nella computazione di Ris* si sarebbero potute eliminare subito le formule che contengono una lettera e la sua negazione perché sono tutte clausole che corrispondono a tautologie e non portano quindi nessun risultato quando si guarda all'insoddisfacibilità; si sarebbero potute poi anche eliminare, per la stessa ragione, le clausole la cui soddisfacibilità è implicata da altre semplificando così un po' i calcoli. Nella pratica (almeno manuale) comunque in genere non si

costruisce Ris* S, ma si va a cercare un albero di derivazione della clausola vuota dalle clausole di partenza.

Considerando l'esempio precedente e lavorando "a mano" potremmo rappresentare la derivazione tramite un albero di derivazione i cui nodi sono clausole e due nodi (clausole) sono collegati ad un altro nodo che rappresenta la loro risolvente.
Un albero di risoluzione del nostro esempio potrebbe essere quello rappresentato di seguito.



L'albero che abbiamo disegnato è un albero lineare, cioè un albero in cui in ogni passo della risoluzione si è utilizzata l'ultima clausola ottenuta. Una derivazione fatta in questo modo si chiama derivazione per **risoluzione lineare.** Si può dimostrare che, se da un insieme di clausole si può ricavare la clausola vuota, allora la clausola vuota si può ottenere anche tramite risoluzione lineare. Quando si lavora tramite risoluzione su un dato insieme di clausole Γ, le clausole occorrenti in Γ si chiamano **clausole di input**. Sarebbe interessante dal punto di vista applicativo poter fare una derivazione per risoluzione lineare che utilizzi la clausola appena ricavata assieme ad una clausola di input in ogni passo della risoluzione (escluso ovviamente il primo passo in cui si utilizzano due clausole di input), questo porterebbe infatti ad un risparmio della memoria necessaria per eseguire la risoluzione. Una risoluzione i cui passi siano fatti in questo modo si chiama **derivazione per risoluzione lineare per input**.

L’albero di risoluzione che abbiamo considerato precedentemente rappresenta un albero di risoluzione lineare, non per input (le clausole racchiuse in rettangoli rossi, ovvero le clausole {~A} e {~C}, non sono di input ma clausole ottenute precedentemente per risoluzione).
In generale la risoluzione lineare per input non è completa, nel senso che non sempre da un insieme insoddisfacibile di clausole è possibile ottenere la clausola vuota tramite risoluzione lineare per input.
La derivazione lineare per input è completa quando l’insieme di clausole di partenza è costituito da **clausole di Horn**, ovvero da clausole che contengono al più un letterale positivo (ovvero al più una lettera enunciativa non negata), questo significa che da un insieme di clausole di Horn possiamo ricavare la clausola vuota se e solo se la possiamo ricavare tramite una risoluzione lineare per input.
In molti casi pratici le clausole che si incontrano sono clausole di Horn.