LA LOGICA DEI PREDICATI DEL PRIMO ORDINE[1]

La logica è quella scienza che fornisce all'uomo gli strumenti indispensabili per controllare con sicurezza la rigorosità dei ragionamenti.

La logica fornisce gli strumenti formali per:

- analizzare le inferenze in termini di operazioni su espressioni simboliche;
- dedurre conseguenze da certe premesse;
- studiare la verità o falsità di certe proposizioni data la verità o falsità di altre proposizioni;
- stabilire la consistenza e la validità di una data teoria.

---

1 Materiale preso dal libro: L. Console, E. Lamma, P. Mello, M.Milano: " Programmazione Logica e Prolog" , Seconda Edizione UTET editore.

LOGICA E INFORMATICA

La logica è utilizzata:

- In Intelligenza Artificiale
  - come linguaggio formale per la rappresentazione di conoscenza
    - semantica non ambigua
    - sistemi formali di inferenza
  - per sistemi di dimostrazione automatica di teoremi e studio di meccanismi efficienti per la dimostrazione

- Per la progettazione di reti logiche;

- Nei database relazionali, come potente linguaggio per l'interrogazione intelligente;

- Come linguaggio di specifica di programmi che per eseguire prove formali di correttezza;

- Come un vero e proprio linguaggio di programmazione (programmazione logica e PROLOG).

LOGICA CLASSICA

Si suddivide in due classi principali:
- ***logica proposizionale***
- ***logica dei predicati.***

Permettono di esprimere proposizioni (cioè frasi) e relazioni tra proposizioni.

La principale differenza tra le due classi è in termini di espressività: nella logica dei predicati è possibile esprimere variabili e quantificazioni, mentre questo non è possibile nella logica proposizionale.

Il linguaggio della logica dei predicati del primo ordine è definito da:
- una ***sintassi***: caratteristiche strutturali del linguaggio formale (mediante una grammatica) senza attribuire alcun significato ai simboli;
- una ***semantica***, che interpreta le frasi sintatticamente corrette del linguaggio. Si dà una interpretazione alle formule stabilendo se una frase è vera o falsa.

SINTASSI DELLA LOGICA DEI PREDICATI

Alfabeto, che consiste di cinque insiemi:

- l'insieme dei simboli di costante, C;
- l'insieme dei simboli di funzione, F;
- l'insieme dei simboli di predicato (o relazione), P;
- l'insieme dei simboli di variabile, V;
- i connettivi logici:

$\sim$ (negazione),
$\wedge$ (congiunzione),
$\vee$ (disgiunzione),
$\leftarrow$ (implicazione),
$\leftrightarrow$ (equivalenza),
le parentesi "(" ")"
e i quantificatori esistenziale ($\exists$) e universale ($\forall$).

Costanti: singole entità del dominio del discorso.
Es. "maria", "giovanna", "3" $\Rightarrow$ iniziale minuscola

Variabili: entità non note del dominio,
Es. X, Y $\Rightarrow$ iniziale maiuscola

Funzioni n-arie: individua univocamente un oggetto del dominio del discorso mediante una relazione tra altri "n" oggetti del dominio.
Es. madre(maria)

Importante: le funzioni, in logica, non presuppongono alcun concetto di valutazione

Predicati n-ari: generica relazione (che può essere vera o falsa) fra "n" oggetti del dominio del discorso.

Es. parente(giovanna,maria)

Date queste definizioni principali possiamo definire:

Termine (definito ricorsivamente):

- una variabile è un termine;
- una costante è un termine;
- se f è un simbolo di funzione n-aria e $t_1,...t_n$ sono termini, allora $f(t_1,...,t_n)$ è un termine.

Es. maria, f(X)

Atomo o formula atomica:
l'applicazione di un simbolo di predicato n-ario p a n termini $t_1,...,t_n$: $p(t_1,..,t_n)$.

Es. parente(giovanna,maria)

Espressione o formula: sequenza di simboli appartenenti all'alfabeto.

Es

parente(giovanna, maria)	(E1)
$\exists$X (uomo(X) $\wedge$ felice(X))	(E2)
$\forall$X (uomo(X) $\rightarrow$ mortale(X))	(E3)
$\exists$X (uomo(X) $\wedge$)	(E4)
$\exists$X (uomo(f(X)	(E5)

Formule ben formate (fbf): frasi sintatticamente corrette del linguaggio. Si ottengono attraverso combinazione di formule atomiche, utilizzando i connettivi e i quantificatori. Sono definite ricorsivamente come segue:

- ogni atomo è una fbf;
- se A e B sono fbf, allora lo sono anche $\sim A$, $A \wedge B$, $A \vee B$, $A \rightarrow B$, $A \leftrightarrow B$ (eventualmente racchiuse tra parentesi tonde bilanciate);
- se A è una fbf e X è una variabile, $\forall X\ A$ e $\exists X\ A$ sono fbf.

Le espressioni (E1), (E2), (E3) sono formule ben formate, mentre non lo sono (E4) e (E5).

Letterale: fbf atomica o la sua negazione. Ad esempio, la formula (E1) è un letterale.

REGOLE DI PRECEDENZA TRA OPERATORI

$\sim \exists \forall$

$\wedge$

$\vee$

$\rightarrow \leftrightarrow$

Esempio

La fbf:

$$a \vee \sim b \wedge \exists X\ c(X) \rightarrow d(X,Y)$$

è equivalente a:

$$(a \vee ((\sim b) \wedge (\exists X\ c(X)))) \rightarrow d(X,Y)$$

fbf in forma normale prenessa disgiuntiva ("disjunctive prenex normal form"): disgiunzione di una o più fbf composte da congiunzioni di letterali; le quantificazioni compaiono tutte in testa a F.

fbf in forma normale prenessa congiuntiva ("conjunctive prenex normal form"): congiunzione di una o più fbf composte da disgiunzioni di letterali; le quantificazioni compaiono tutte in testa ad F.

Esempio
La fbf:

$\exists X \forall Y \exists Z (a(X) \wedge b(Y,Z)) \vee (c(X) \wedge \sim a(Z) \wedge d) \vee f$

è in forma normale disgiuntiva.
La fbf:

$\exists X \forall Y \exists Z (a(X) \vee b(Y,Z)) \wedge (c(X) \vee \sim a(Z) \vee d) \wedge f$

è in forma normale congiuntiva.

Qualunque fbf può essere trasformata in forma normale prenessa (congiuntiva o disgiuntiva) attraverso opportune trasformazioni sintattiche.

Campo di azione (***scope***) di un quantificatore: fbf che lo segue immediatamente. Nel caso di ambiguità si utilizzano le parentesi tonde.

Esempio
Nella fbf: $\forall X (p(X,Y) \wedge q(X)) \vee q(X)$

la quantificazione sulla variabile X ha come campo d'azione la formula $p(X,Y) \wedge q(X)$.

Variabili libere: variabili che non compaiono all'interno del campo di azione di un quantificatore.

Esempio

Nella fbf: $F = \forall X\ (p(X,Y) \wedge q(X))$
la variabile Y risulta libera in F.

Formule chiuse: fbf che non contengono alcuna variabile libera. Ad esempio, le formule (E1), (E2) ed (E3) sono fbf chiuse. Nel seguito considereremo solo formule fbf chiuse.

Formule ground: formule che non contengono variabili. Ad esempio la formula (E1) è una formula "ground".

Varianti: una formula F2, ottenuta rinominando le variabili di una formula F1, è detta *variante* di F1.

Esempio

La formula:

$$\forall X\ \exists Y\ p(X,Y)$$

è una variante della formula $\forall W\ \exists Z\ p(W,Z)$.

SEMANTICA

Occorre associare un significato ai simboli.
Ogni sistema formale è la modellizzazione di una certa realtà (ad esempio la realtà matematica).

Un'*interpretazione* è la costruzione di un rapporto fra i simboli del sistema formale e tale realtà (chiamata anche *dominio del discorso*).

Ogni formula atomica o composta della logica dei predicati del primo ordine può assumere il valore vero o falso in base alla frase che rappresenta nel dominio del discorso.

Esempio:

$$\forall X \, \forall Y \, \forall Z \;\; (\mathrm{op}(X, Y, Z) \rightarrow \mathrm{op}(Y, X, Z))$$

se X, Y, Z variano sull'insieme dei numeri reali tale formula è vera se il simbolo di predicato "op" ha il significato di un operatore commutativo (es: somma o moltiplicazione), falsa se l'operatore non è commutativo (es. sottrazione o divisione).

INTERPRETAZIONE

Dato un linguaggio del primo ordine L un'*interpretazione* per L definisce un dominio non vuoto D e assegna:

- a ogni simbolo di costante in C, una costante in D;
- a ogni simbolo di funzione n-ario F, una funzione: $F: D^n \rightarrow D$;
- a ogni simbolo di predicato n-ario in P una relazione in D^n, cioè un sottoinsieme di D^n.

Esempio
Linguaggio del primo ordine, L, nel quale si ha una costante "0", un simbolo di funzione unaria "s" e un simbolo di predicato binario "p".

Interpretazione I1 D: numeri naturali.
"0" rappresenta il numero zero.
"s" rappresenta il successore di un numero naturale
"p" rappresenta la relazione binaria "≤"

Interpretazione I1 D: numeri interi negativi.
"0" rappresenta il numero zero.
"s" rappresenta il predecessore di un numero naturale
"p" rappresenta la relazione binaria "≤"

VALORE DI VERITÀ DI UNA FBF

Data un'interpretazione il *valore di verità* di una fbf si definisce secondo le seguenti regole.

1) Formula atomica "ground" ha valore vero sotto un'interpretazione quando il corrispondente predicato è soddisfatto (cioè quando la corrispondente relazione è vera nel dominio). La formula atomica ha valore falso quando il corrispondente predicato non è soddisfatto.

Esempio
Interpretazione I1. p(0,s(0)) vero
p(s(0), 0) falso
Interpretazione I2. p(0,s(0)) falso
p(s(0), 0) vero

2) Formula composta il valore di verità di una formula composta rispetto a un'interpretazione si ottiene da quello delle sue componenti utilizzando **le tavole di verità** dei connettivi logici.

A	B	~A	$A \wedge B$	$A \vee B$	$A \Rightarrow B$	$A \Leftrightarrow B$
T	T	F	T	T	T	T
T	F	F	F	T	F	F
F	T	T	F	T	T	F
F	F	T	F	F	T	T

Nota: l'implicazione $A \Rightarrow B$ è diversa rispetto al "se ..... allora" utilizzato nel linguaggio naturale.
A: antecedente B: conseguente

Esempi:

Data la formula F:

volano(asini) $\Rightarrow$ ha_scritto(manzoni,promessi_sposi)

assumendo l'interpretazione più intuitiva, F ha valore vero, poiché l'antecedente ha valore falso in tale interpretazione.

La formula F:

$$p(s(0),0) \Rightarrow p(0,s(0))$$

ha valore vero nell'interpretazione I1 poiché l'antecedente ha valore falso, mentre ha valore falso in I2 poiché a un antecedente vero corrisponde un conseguente falso.

3) Formula quantificata esistenzialmente: una formula del tipo ∃X F è vera in un'interpretazione I se esiste almeno un elemento d del dominio D tale che la formula F', ottenuta assegnando d alla variabile X, è vera in I. In caso contrario F ha valore falso.

Esempio

La formula ∃X p(X,s(0)) ha valore vero nell'interpretazione I1 in quanto esiste un numero naturale, zero, minore di uno, tale che la formula F'=p(0,s(0)) ha valore vero in I1.

4) Formula quantificata universalmente: una formula del tipo ∀X F è vera in un'interpretazione I se per ogni elemento d del dominio D, la formula F', ottenuta da F sostituendo d alla variabile X, è vera in I. Altrimenti F ha valore falso.

Esempio

La fbf ∀Y p(0,Y) ha valore vero rispetto alle interpretazioni I1 (dove viene interpretata come "0 è minore o uguale a ogni intero positivo Y"), mentre ha valore falso rispetto a I2 poiché esiste almeno un elemento del dominio che la falsifica (esempio non è vero che "0 è minore o uguale a –1").

MODELLI

Data una interpretazione I e una fbf chiusa F, I è un ***modello*** per F se e solo se F è vera in I.

Esempio
Per la fbf $\forall Y\ p(0,Y)$ l'interpretazione I1 è un modello, mentre I2 non lo è.

Una fbf è ***soddisfacibile*** se e solo se è vera almeno in una interpretazione, ovvero se esiste almeno un modello per essa.

Una fbf che ha valore vero per tutte le possibili interpretazioni, cioè per cui ogni possibile interpretazione è un modello, è **detta *logicamente valida***.

Esempio
La fbf: $\forall X\ p(X) \vee \sim(\forall Y\ p(Y))$ è logicamente valida. Infatti, le formule $\forall X\ p(X)$ e $\forall Y\ p(Y)$ sono semplici varianti della stessa formula F e quindi hanno i medesimi valori di verità per qualunque interpretazione. In generale, $F \vee \sim F$ ha sempre valore vero, in modo indipendente dall'interpretazione.

F logicamente valida $\Leftrightarrow$ ~F è non soddisfacibile.
F è soddisfacibile $\Leftrightarrow$ ~F non è logicamente valida.

INSIEMI DI FORMULE

Un insieme di formule chiuse del primo ordine S è ***soddisfacibile*** se esiste una interpretazione I che soddisfa tutte le formule di S (cioè che è un modello per ciascuna formula di S). Tale interpretazione è detta modello di S.

Esempio
Si consideri il seguente insieme di formule S:

$$S=\{\forall Y\ p(Y,Y),\ p(s(0),0) \Rightarrow p(0,s(0))\}.$$

L'interpretazione I1 è modello di S, mentre I2 non lo è. In I2 è infatti soddisfatta la prima formula dell'insieme, ma non la seconda.

Un insieme di formule S che non può essere soddisfatto da alcuna interpretazione, è detto ***insoddisfacibile*** (o inconsistente). Ad esempio l'insieme di formule {A,~A} è insoddisfacibile.

Esempio
Esempi di insiemi di formule insoddisfacibili sono:

$$S1=\{\sim (\exists X\ \forall Y\ p(X,Y)),\ \exists X\ \forall Y\ p(X,Y)\}$$

$$S2=\{p(s(0),0)\Rightarrow p(0,s(0)),\ p(s(0),0),\ \sim p(0,s(0))\}$$

In S1, infatti, compaiono una formula e la sua negazione. In S2, per ogni interpretazione in cui p(s(0),0) e ~p(0,s(0)) sono vere, la formula $p(s(0),0) \Rightarrow p(0,s(0))$ non è vera, per la tabella di verità della negazione e dell'implicazione.

CONSEGUENZA LOGICA

Una formula F ***segue logicamente*** (o è ***conseguenza logica***) da un insieme di formule S (e si scrive **S |= F**), se e solo se ogni interpretazione I che è un modello per S, è un modello per F.

Esempio
Si consideri l'insieme di fbf S:
{p(0,0), $\forall$X p(X,X), $\forall$X $\forall$Y (p(X,Y) $\Rightarrow$p(X,s(Y)))}
Da S segue logicamente la formula F=p(0,s(0)) poiché ogni interpretazione I che soddisfa S soddisfa anche F.

Dall'insieme S, invece, non segue logicamente la formula F1: p(s(0),0) in quanto esiste un'interpretazione (I1) che soddisfa S, ma non F1.

Proprietà $\frac{\textit{Condizioni}}{\textit{Conclusione}}$

Se una fbf F segue logicamente da S (S |= F), allora l'insieme S $\cup$ {~F} è insoddisfacibile.
Viceversa, se S $\cup$ {~F} è insoddisfacibile (e S era soddisfacibile), allora F segue logicamente da S.

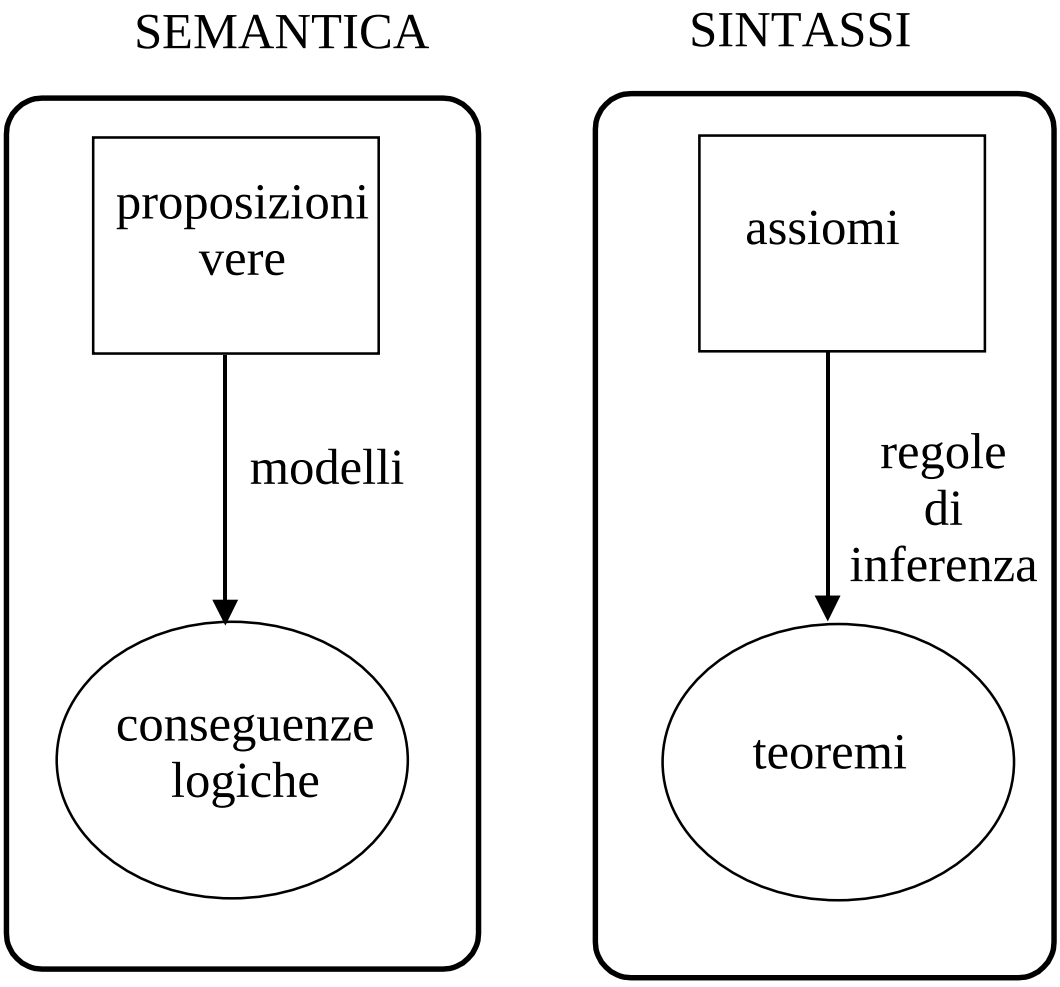
Difficile lavorare a livello semantico (interpretazione, modelli). Quindi si lavora a livello sintattico.

SISTEMI DI REFUTAZIONE

I sistemi di refutazione si basano su questa proprietà: per dimostrare S |= F supposto S soddisfacibile è sufficiente dimostrare che S∪{~F} è insoddisfacibile.

Problema interessante:

Determinare se una formula F segue logicamente da S (ovvero che S ∪ {~F} è insoddisfacibile) utilizzando solo semplici trasformazioni sintattiche ***(regole di inferenza)***, possibilmente ripetitive e quindi automatizzabili, e non introducendo concetti quali significato o interpretazione o modello.



TEORIE DEL PRIMO ORDINE

Calcolo proposizionale: verifica di formula/e vera/e tramite le tavole di verità
Calcolo dei predicati del primo ordine: tavole di verità troppo complesse. Dominio di interpretazione estremamente grande, se non infinito. Si ricorre al metodo assiomatico (noto come ***proof theory***).
A questo scopo, la logica dei predicati del primo ordine può essere formulata come ***sistema assiomatico-deduttivo***.

Teoria assiomatica

- formule ben formate ritenute vere: *assiomi*
- criteri di manipolazione sintattica: *regole di inferenza* derivano fbf da fbf

$$\frac{\text{condizioni}}{\text{conclusione}}$$

Scopo: produrre nuove formule sintatticamente corrette (***teoremi***).

Semplificazioni:
Consideriamo solo i connettivi logici ~ (negazione) e $\rightarrow$ (implicazione). Gli derivano da questi. VERIFICA

$(A \wedge B)$ equivale a $(\sim(A \rightarrow (\sim B)))$

$(A \vee B)$ equivale a $((\sim A) \rightarrow B)$

$(A \times B)$ equivale a $((A \rightarrow B) \wedge (B \rightarrow A))$

Inoltre, per i quantificatori:

$\exists X\ A$ abbrevia $\sim(\forall X\ \sim A)$

$\forall X\ A$ abbrevia $\sim(\exists X\ \sim A)$

REGOLE DI INFERENZA DEL CALCOLO DEI PREDICATI

Modus Ponens (MP):

$$\frac{A, A \rightarrow B}{B}$$

che deriva da due formule del tipo A e $A \rightarrow B$ la nuova formula B.

Specializzazione (Spec): $\dfrac{\forall X \quad A}{A(t)}$

Da una formula quantificata universalmente è possibile derivare una formula identica all'originale in cui la variabile X è sostituita da un elemento del dominio del discorso (costante e funzione).

DIMOSTRAZIONE DI TEOREMI

Dimostrazione: sequenza finita di fbf f_1, f_2, ..., f_n, tale che ciascuna f_i o è un assioma oppure è ricavabile dalle fbf precedenti mediante una regola di inferenza.

Teorema: L'ultima fbf di ogni dimostrazione.

Prova del teorema: sequenza di regole di inferenza applicate.

Una fbf F è ***derivabile*** in una teoria T (**T |- F**) se esiste una sequenza di fbf f_1, f_2, ..., f_n, tale che f_n = F e, per ogni i, o f_i è un assioma di T, oppure è ricavabile dalle fbf precedenti mediante una regola di inferenza di T.

Esempio

Teoria T: assiomi propri (relazione di minore uguale sui numeri naturali):

p(0,0)	(A1)
$\forall X \forall Y (p(X,Y) \Rightarrow p(X,s(Y)))$	(A2)
$\forall X$ p(X,X)	(A3)

Teorema p(0,s(0)) (cioè T |- p(0,s(0)))

Trasformazione da Spec e A2:

$\forall X \forall Y (p(X,Y) \Rightarrow p(X,s(Y))) \Rightarrow \forall Y (p(0,Y) \Rightarrow p(0,s(Y)))$ (T1)

applicando MP a T1 e A2: $\forall Y (p(0,Y) \Rightarrow p(0,s(Y)))$ (T2)

da Spec. e T2:

$\forall Y(p(0,Y) \Rightarrow p(0,s(Y))) \Rightarrow p(0,0) \Rightarrow p(0,s(0)))$ (T3)

applicando MP a T3 e T2: $p(0,0) \Rightarrow p(0,s(0))$ (T4)

applicando MP a T4 e A1: p(0,s(0)) (T5)

DECIDIBILITÀ

Teoria decidibile teoria per la quale esiste un metodo meccanico per stabilire se una qualunque fbf è un teorema o non lo è.

Il calcolo dei predicati del primo ordine non è decidibile, ma ***semi-decidibile***: se una formula è un teorema, esiste un metodo meccanico che la deriva in un numero finito di passi. Se invece la formula non è un teorema, non è garantita, in generale, la terminazione del metodo meccanico.

Una teoria del primo ordine è un insieme di fbf chiuse (assiomi) e si può quindi parlare di *modello di una teoria*.

Un modello per una teoria del primo ordine T è un'interpretazione che soddisfa tutti gli assiomi di T (assiomi logici e assiomi propri).

Se T ha almeno un modello viene detta *consistente* (o *soddisfacibile*).

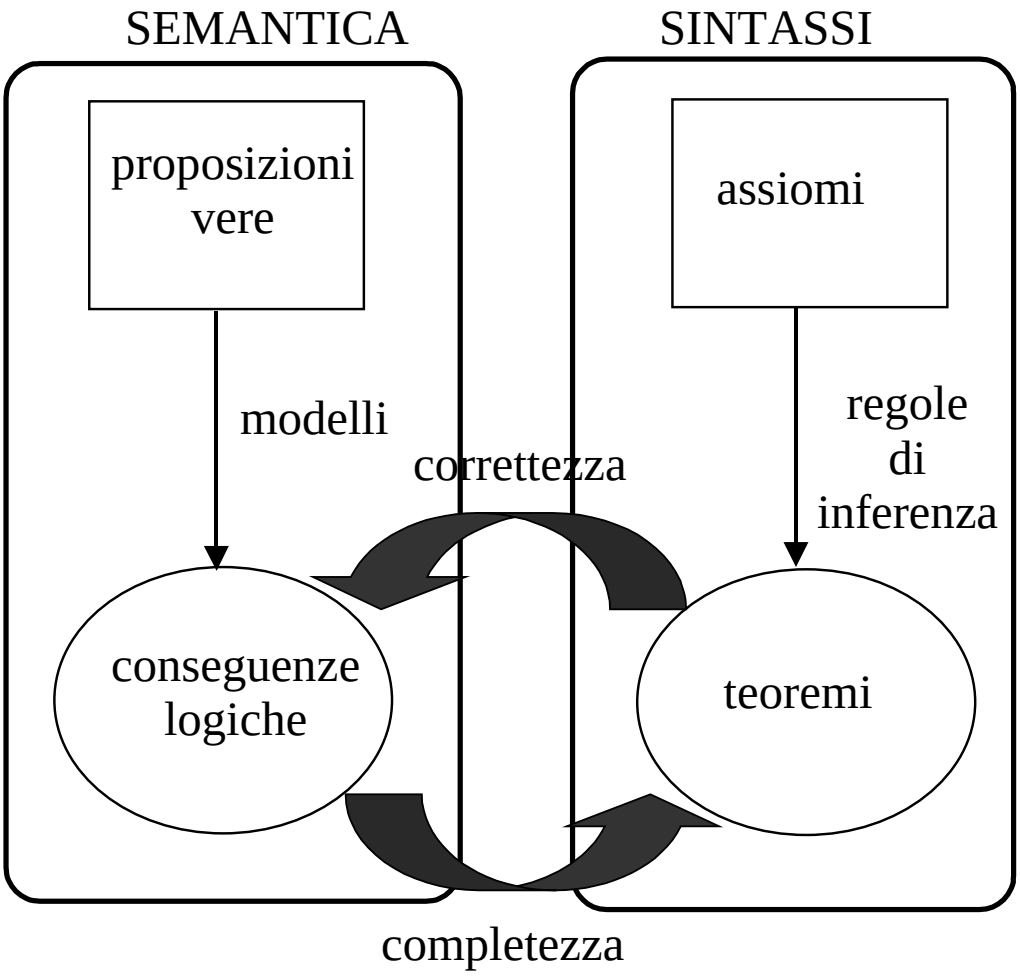
CORRETTEZZA E COMPLETEZZA

Una teoria assiomatica è ***corretta*** se i teoremi dimostrati seguono logicamente dagli assiomi della teoria.

Una teoria assiomatica è ***completa*** se tutte le fbf che seguono logicamente dalla teoria possono essere dimostrati come teoremi della teoria.

Se T è corretta e completa è garantita l'equivalenza tra l'aspetto sintattico e semantico

T |- F ⇔ T|= F.



Esempio

Si consideri una teoria del primo ordine T, data dai seguenti assiomi propri che rappresentano la relazione di minore sui numeri naturali:

p(0,s(0)) (A1)

$\forall$ X $\forall$ Y (p(X,Y)$\rightarrow$p(X,s(Y))) (A2)

$\forall$X p(X,s(X)) (A3)

Le regole di inferenza di T siano Modus Ponens, Specializzazione e la seguente regola:

Abduzione (ABD):

$$\frac{B, A \rightarrow B}{A}$$

In T si deriva come teorema la formula p(0,0) applicando le seguenti trasformazioni:

da Spec. e A2:

$\forall$X$\forall$Y(p(X,Y) $\rightarrow$p(X,s(Y)))$\rightarrow\forall$Y (p(0,Y) $\rightarrow$p(0,s(Y))) (T1)

applicando MP a T1 e A2:

$\forall$Y (p(0,Y) $\rightarrow$ p(0,s(Y))) (T2)

da Spec. e T2:

$\forall$Y (p(0,Y) $\rightarrow$p(0,s(Y))) $\rightarrow$ (p(0,0) $\rightarrow$ p(0,s(0))) (T3)

applicando MP a T3 e T2:

p(0,0) $\rightarrow$ p(0,s(0)) (T4)

applicando ABD a T4 e A6:

p(0,0) (T5)

A causa dell'applicazione dell'abduzione, questa teoria **non è corretta**: un'interpretazione che ha come dominio l'insieme dei numeri naturali e associa al simbolo di funzione "s" la funzione successore e al simbolo di predicato "p" la relazione < (minore) è un modello per gli assiomi, ma non per la formula p(0,0).

Esempio
sta-male(mario).
$\forall$ X (ha-epatite(X) $\rightarrow$ sta-male(X)).

si conclude:
ha-epatite(mario).

ERRORE !!

Abduzione, esempi:

$\forall$ X (person(X) $\rightarrow$ mortal(X)).
mortal(tweety).

Allora deriviamo: person(tweety).

Vincoli: $\forall$ X not(person(X) and bird(X)).

Se aggiungiamo: bird(tweety)
violiamo i vincoli.

Esempio
Ragionamento abduttivo usato per ***diagnosi*** di guasti

Teoria:
ruota_traballante:- raggi_rotti.
ruota_traballante:- gomma_sgonfia.
gomma_sgonfia:- valvola_difettosa.
gomma_sgonfia:- forata_camera_aria.
gomma_mantiene_aria.

Vincoli:
:- gomma_sgonfia, gomma_mantiene_aria

Goal
?- ruota_traballante.

Risposta: yes if raggi_rotti

Mentre:
    yes if valvola_difettosa
    yes if forata_camera_aria

non sono accettabili in quanto violano i vincoli.

MONOTONICITÀ

Un'altra proprietà fondamentale delle teorie del primo ordine è la *monotonicità*. Una teoria T è monotona se l'aggiunta di nuovi assiomi non invalida i teoremi trovati precedentemente.

Proprietà
Sia Th(T) l'insieme dei teoremi derivabili da T. Allora T è monotona se $Th(T) \subseteq Th(T \cup H)$ per qualunque insieme aggiuntivo di assiomi H.

Esistono regole di inferenza non monotone. Ad esempio la regola nota come *Assunzione di Mondo Chiuso* ("*Closed World Assumption*"):

Assunzione di Mondo Chiuso (CWA):

$$\frac{T \not\vdash A}{\sim A}$$

se una formula atomica "ground" A non è conseguenza logica di una teoria T, ~A si può considerare un teorema di T. Se alla teoria T si aggiunge l'assioma A, non si può più derivare ~A, da cui segue la non monotonicità del sistema di inferenza.

IL PRINCIPIO DI RISOLUZIONE

Sistema di deduzione per la logica a clausole per il quale valgono interessanti proprietà.

Regola di inferenza**:** ***Principio di Risoluzione*** [Robinson 65], che si applica a teorie del primo ordine in ***forma a cla****usole*.

È la regola di inferenza base utilizzata nella programmazione logica.

CLAUSOLE

Una ***clausola*** è una disgiunzione di *letterali* (cioè formule atomiche negate e non negate), in cui tutte le variabili sono quantificate universalmente in modo implicito.

Una clausola generica può essere rappresentata come la disgiunzione:

$$A_1 \vee A_2 \vee ... \vee A_n \vee \sim B_1 \vee ... \vee \sim B_m$$

dove A_i (i=1,..,n) e B_j (j=1,..,m) sono atomi.

Una clausola nella quale non compare alcun letterale, sia positivo sia negativo, è detta *clausola vuota* e verrà indicata con . interpretato come contraddizione: disgiunzione falso $\vee$ ~vero

Un sottoinsieme delle clausole è costituito dalle ***clausole definite***, nelle quali si ha sempre un solo letterale positivo:

$$A_1 \vee \sim B_1 \vee \ldots \vee \sim B_m$$

TRASFORMAZIONE IN CLAUSOLE

Passi per trasformare una qualunque fbf della logica dei predicati del primo ordine in un insieme di clausole

1) ***Trasformazione in fbf chiusa***

Esempio la formula:

$$\forall X\ (p(Y) \rightarrow \sim(\forall Y\ (q(X,Y) \rightarrow p(Y)))) \qquad (1)$$

è trasformata in:

$$\forall X\ \forall Y\ (p(Y) \rightarrow \sim(\forall Y\ (q(X,Y) \rightarrow p(Y)))) \quad (2)$$

2) ***Applicazione delle equivalenze per i connettivi logici*** (ad esempio $A \rightarrow B$ è sostituito da $\sim A \vee B$) e la si riduce in forma *and-or*.

La formula (2) diventa:

$$\forall X\ \forall Y\ (\sim p(Y) \vee \sim(\forall Y\ (\sim q(X,Y) \vee p(Y)))) \quad (3)$$

3) ***Applicazione della negazione ad atomi e non a formule composte***, tenendo presente che:

$\forall X \sim A$ equivale a $\sim \exists X\ A$

$\exists X \sim A$ equivale a $\forall X \sim A$

$\sim(A1 \vee A2 \vee ... \vee An)$ equivale a $\sim A1 \wedge \sim A2 \wedge ... \wedge \sim An$

$\sim(A1 \wedge A2 \wedge ... \wedge An)$ equivale a $\sim A1 \vee \sim A2 \vee ... \vee \sim An$

(leggi di De Morgan).

(3) si modifica in:

$$\forall X\ \forall Y\ (\sim p(Y) \vee (\exists Y\ (q(X,Y) \wedge \sim p(Y)))) \qquad (4)$$

4) ***Cambiamento di nomi delle variabili***, nel caso di conflitti.
in (4) la seconda variabile Y viene rinominata Z:

$$\forall X \forall Y (\sim p(Y) \vee (\exists Z (q(X,Z) \wedge \sim p(Z)))) \quad (5)$$

5) ***Spostamento dei quantificatori*** in testa alla formula (*forma prenessa*).

$$\forall X \forall Y \exists Z (\sim p(Y) \vee (q(X,Z) \wedge \sim p(Z))) \quad (6)$$

6) ***Forma normale congiuntiva*** cioè come congiunzione di disgiunzioni, con quantificazione in testa.

$$\forall X \forall Y \exists Z ((\sim p(Y) \vee q(X,Z)) \wedge (\sim p(Y) \vee \sim p(Z))) \quad (7)$$

7) ***Skolemizzazione:*** ogni variabile quantificata esistenzialmente viene sostituita da una funzione delle variabili quantificate universalmente che la precedono. Tale funzione è detta *funzione di Skolem*.

Ad esempio una formula del tipo: $\forall X \exists Y p(X,Y)$ può essere espressa in modo equivalente come: $\forall X p(X,g(X))$
In (7) Z è sostituita da f(X,Y), perché Z si trova nel campo di azione delle quantificazioni $\forall X$ e $\forall Y$:

$$\forall X \forall Y ((\sim p(Y) \vee q(X,f(X,Y))) \wedge (\sim p(Y) \vee \sim p(f(X,Y)))) \quad (8)$$

Perdita in espressività. Non è la stessa cosa asserire: F: $\exists X p(X)$ oppure F': p(f).
Vale comunque la proprietà che F è inconsistente se e solo se F' è inconsistente.

8) ***Eliminazione dei quantificatori universali***: si ottiene è una formula detta *universale* (tutte le sue variabili sono quantificate universalmente) in forma normale congiuntiva.

$$((\sim p(Y) \vee q(X,f(X,Y))) \wedge (\sim p(Y) \vee \sim p(f(X,Y)))) \quad (9)$$

Una formula di questo tipo rappresenta ***un insieme di clausole*** (ciascuna data da un congiunto nella formula). La forma normale a clausole che si ottiene:

$$\{\sim p(Y) \vee q(X,f(X,Y)), \sim p(Y) \vee \sim p(f(X,Y))\} \quad (10)$$

La seconda clausola può essere riscritta rinominando le variabili (sostituendo cioè la formula con una sua *variante*).

$$\{\sim p(Y) \vee q(X,f(X,Y)), \sim p(Z) \vee \sim p(f(W,Z))\} \quad (11)$$

Qualunque teoria del primo ordine T può essere trasformata in una teoria T' in forma a clausole.
Anche se T non è logicamente equivalente a T' (a causa dell'introduzione delle funzioni di Skolem), vale comunque la seguente proprietà:

Proprietà
Sia T una teoria del primo ordine e T' una sua trasformazione in clausole. Allora T è insoddisfacibile se e solo se T' è insoddisfacibile.

Il principio di risoluzione è una procedura di dimostrazione che opera per contraddizione e si basa sul concetto di insoddisfacibilità.

IL PRINCIPIO DI RISOLUZIONE

Il *principio di risoluzione*, che si applica a formule in forma a clausole, è molto più efficiente del metodo assiomatico-deduttivo ed è utilizzato dalla maggior parte dei risolutori automatici di teoremi.

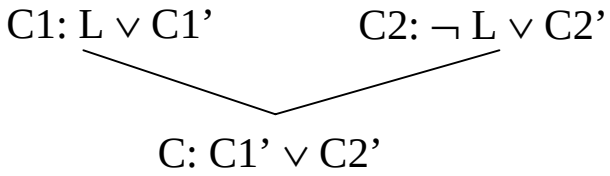
Logica Proposizionale: clausole prive di variabili.
Siano C_1 e C_2 due clausole prive di variabili:

$C_1 = A_1 \vee ... \vee A_n$ $\qquad$ $C_2 = B_1 \vee ... \vee B_m$

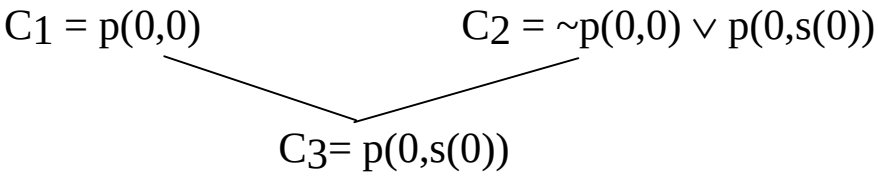
Se esistono in C_1 e C_2 due letterali ***opposti***, A_i e B_j, ossia tali che $A_i = \sim B_j$, allora da C_1 e C_2, (clausole ***parent***) si può derivare una nuova clausola C_3, denominata ***risolvente***, della forma:

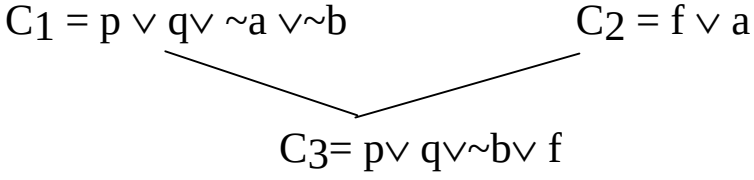
$C_3 = A_1 \vee .. \vee A_{i-1} \vee A_{i+1} \vee .. \vee A_n \vee B_1 \vee .. \vee B_{j-1} \vee B_{j+1} \vee ... \vee B_m$

C_3 è conseguenza logica di $C_1 \cup C_2$.



Esempi





UNIFICAZIONE

Per applicare il principio di risoluzione alle clausole non “ground” è necessario introdurre il concetto di *unificazione* [Robinson 65].

Unificazione: procedimento di manipolazione formale utilizzato per stabilire quando due espressioni possono coincidere procedendo a opportune *sostituzioni*.

Sostituzione: σ insieme di legami di termini T_i a simboli di variabili X_i (i=1,..,n).

$$\sigma = \{X_1/T_1, X_2/T_2,...,X_n/T_n\}$$

dove $X_1, X_2,..., X_n$ sono distinte.

La sostituzione corrispondente all'insieme vuoto è detta ***sostituzione identità*** (ε).

Applicazione della sostituzione σ a un’espressione E, $[E]\sigma$, produce una nuova espressione ottenuta sostituendo simultaneamente ciascuna variabile X_i dell’espressione con il corrispondente termine T_i. $[E]\sigma$ è detta ***istanza*** di E.

Renaming: sostituzioni che cambiano semplicemente il nome ad alcune delle variabili di E. , $[E]\sigma$ è una ***variante*** di E.

Esempio

L'applicazione della sostituzione $\sigma = \{X/c, Y/a, Z/W\}$ all'espressione p(X,f(Y),b,Z) produce l'istanza p(c,f(a),b,W).

Analogamente:

[c(Y,Z)]{Y/T, Z/neri} = c(T,neri) (e1)
[c(T,neri)]{Y/T, Z/neri} = c(T,neri) (e2)
[c(Y,Z)]{Y/bianchi,T/bianchi,Z/neri}= c(bianchi,neri) (e3)
[c(T,neri)]{Y/bianchi,T/bianchi,Z/neri}=c(bianchi,neri) (e4)
[c(Y,Z)]{Y/T, Z/bianchi} = c(T,bianchi) (e5)
[c(T,neri)]{Y/T, Z/bianchi} = c(T,neri) (e6)
[t(l(a),t(l(b),l(H)))]{H/l(a), Y/l(b), Z/l(l(a))}= t(l(a),t(l(b),l(l(a)))) (e7)
[t(H,t(Y,Z))]{H/T,Y/X,Z/W}=t(T,t(X,W)) (e8)
(e8) è una variante dell'espressione di partenza.
[c(T,neri)]ε = c(T,neri) (e9)
[p(X,Y)]{X/a,Y/X} = p(a,X) (e10)

Combinazioni di sostituzioni:

$\sigma_1=\{X_1/T_1,X_2/T_2,...,X_n/T_n\}$ $\sigma_2=\{Y_1/Q_1,Y_2/Q_2,...,Y_m/Q_m\}$

composizione $\sigma_1\sigma_2$ di σ_1 e σ_2 è la sostituzione

$\{X_1/[T_1]\sigma_2,...,X_n/[T_n]\sigma_2,Y_1/Q_1,Y_2/Q_2,..,Y_m/Q_m\}$

cancellando le coppie $X_i/[T_i]\sigma_2$ per le quali si ha $X_i=[T_i]\sigma_2$ e le coppie Y_j/Q_j per le quali Y_j appartiene all'insieme $\{X_1, X_2,...,X_n\}$.

Esempio

$\sigma_1 = \{X/f(Z), W/R, S/c\}$ $\sigma_2 = \{Y/X, R/W,Z/b\}$

produce: $\sigma_3 = \sigma_1\sigma_2 = \{X/f(b), S/c, Y/X, R/W, Z/b\}$.

Sostituzioni più generali: Una sostituzione θ è *più generale* di una sostituzione σ se esiste una sostituzione λ tale che $\sigma=\theta\lambda$.

Esempio
La sostituzione θ = {Y/T, Z/neri} è più generale della sostituzione σ = {Y/bianchi, T/bianchi, Z/neri} in quanto σ si può ottenere attraverso la composizione {Y/T, Z/neri}{T/bianchi} ($\sigma=\theta\lambda$, e λ = {T/bianchi}).

L'***unificazione*** rende identici due o più atomi (o termini) (o meglio le loro ***istanze***) attraverso un'opportuna sostituzione.

Se si considerano solo due atomi (o termini), uno dei quali senza alcuna variabile, si ricade in un caso particolare di unificazione, detto ***pattern-matching***.

Un insieme di atomi (o termini) A_1, A_2,..., A_n è ***unificabile*** se esiste una sostituzione σ tale che:

$$[A_1]\sigma = [A_2]\sigma = ... = [A_n]\sigma.$$

La sostituzione σ è detta ***sostituzione unificatrice*** (o *unificatore*)

Esempio Se si considerano gli atomi:

A1=c(Y,Z) A2=c(T,neri)

possibili sostituzioni unificatrici sono:

θ = {Y/T, Z/neri}

σ = {Y/bianchi, T/bianchi, Z/neri}

la loro applicazione produce la stessa istanza:

[c(Y,Z)]θ = [c(T,neri)]θ = c(T,neri)

[c(Y,Z)]σ = [c(T,neri)]σ = c(bianchi,neri)

La sostituzione: λ={Y/T, Z/bianchi} non è un unificatore per A1 e A2 perchè produce istanze diverse.

Esempio Per gli atomi:

A3: p(X,X,f(a,Z)) A4: p(Y,W,f(Y,J))

possibili sostituzioni unificatrici sono:

μ = {X/a, Y/a, W/a, J/Z}

φ = {X/a,Y/a,W/a, J/c, Z/c}

la loro applicazione ad A3 e A4 produce la stessa istanza:

p(a,a,f(a,Z)) nel caso della sostituzione μ

p(a,a,f(a,c)) nel caso della sostituzione φ.

Possono esistere più sostituzioni unificatrici, si vuole individuare quella più generale (*mgu*, ***most general unifier***).

μ mgu: φ si ottiene da μ componendola con: α = {Z/c}.

Esempi

θ indica la sostituzione unificatrice più generale:

1) p(X,f(a)) p(b, Y)

   θ = {X/b, Y/f(a)}

2) t(l(a),t(l(b),l(c))) t(l(X),Z)

   θ = {X/a, Z/t(l(b),l(c))}

   I termini t(l(a),t(l(b),l(c))) e t(l(X),Z) rappresentano alberi binari.

3) t(l(a),t(l(b),l(H))) t(H,t(Y,Z))

   θ = {H/l(a), Y/l(b), Z/l(l(a))}

4) f(X,g(Y),T) f(c(a,b),g(g(a,c)),Z).

   θ = {X/c(a,b), Y/g(a,c), T/Z}

5) .(a,.(b,.(c,.(d,[])))) .(X,Y)

   θ = {X/a, Y/.(b,.(c,.(d,[])))}

   rappresentano liste

6) .(a,.(b,.(c,.(d,[])))) .(X,.(Y,Z))

   θ = {X/a, Y/b, Z/.(c,.(d,[]))}

7) .(a,.(b,.(c,.(d,[])))) .(c,Y)

   Non sono unificabili.

8) t(t(t(X,Y),l(c)),Y) t(t(t(Y,Y),l(c)),l(X))

   Non sono unificabili

ALGORITMO DI UNIFICAZIONE:

Algoritmo in grado di determinare se due atomi (o termini) sono unificabili o meno e restituire, nel primo caso, la sostituzione unificatrice più generale.

Esistono vari algoritmi di unificazione di differente complessità.

Regole alla base dell'algoritmo di unificazione fra due termini T1 e T2.

T1 \ T2	<costante> C2	<variabile> X2	<termine composto> S 2
<costante> C1	ok se C1=C2	ok {X2/C1}	NO
<variabile> X1	ok {X1/C2}	ok {X1/X2}	ok {X1/S2}
<termine composto> S1	NO	ok {X2/S1}	ok se S1 e S2 hanno stesso funtore e arietà e gli argomenti UNIFICANO

Un semplice algoritmo di unificazione in un linguaggio "pseudo-Pascal":

Funzione UNIFICA che ha come parametri di ingresso due atomi o termini da unificare A e B e la sostituzione SOST eventualmente già applicata.

La funzione termina sempre ed è in grado di fornire o la sostituzione più generale per unificare A e B o un fallimento (FALSE).

Un termine composto (cioè diverso da costante o variabile) è rappresentato da una lista che ha come primo elemento il simbolo funzionale del termine e come altri elementi gli argomenti del termine.

Es: f(a,g(b,c),X) rappresentato come: [f,a,[g,b,c],X].

La funzione head, applicata a una lista L, restituisce il primo elemento di L, mentre la funzione tail il resto della lista L.

Nell'esempio, head([f,a,[g,b,c],X]) restituisce f, mentre tail([f,a,[g,b,c],X]) restituisce [a,[g,b,c],X].

```
function UNIFICA
(A,B:termini; SOST:sostituzione): sostituzione;
var a,b:termini;
begin
  if  SOST = false
  then  UNIFICA:= false
  else
      begin
          a := [ A ] SOST;
          b := [ B ] SOST;
          if <a e b  sono costanti e a=b>
          then UNIFICA:=SOST
          else
  if <a è una variabile e non è presente in b >
              then
                  begin
                      <aggiorna SOST con la
                  nuova sostituzione  {a/b} >;
                      UNIFICA:=SOST
                  end
              else
                  if   <b è una variabile e
                       non è presente in a >
                  then
                       begin
              <aggiorna SOST con la
              nuova sostituzione  {b/a} >;
                            UNIFICA:=SOST
                       end
                  else
              if <a e b sono termini composti>
      then UNIFICA:=UNIFICA(head(a),head(b),
UNIFICA(tail(a),tail(b),SOST)
                       else UNIFICA:=  FALSE
      end
end;
```

OCCUR CHECK

È il controllo che un termine variabile da unificare con un secondo termine non compaia in quest'ultimo. Necessario per assicurare la terminazione dell'algoritmo e la correttezza del procedimento di unificazione.

I due termini "X" e "f(X)" non sono: non esiste una sostituzione per X che renda uguali i due termini.

Se un termine t ha una struttura complessa, la verifica se X compare in t può essere anche molto inefficiente.

Prolog NON utilizza l'occur-check: non corretto !!

Esempio: Si consideri il programma:

P4 `p(X,f(X)).`

e la query `?- p(Y,Y).`
Per Prolog `p(Y,Y)` segue logicamente da P4 e viene prodotta la sostituzione (contenente un termine infinito): `Y=f(f(f(.....)))`

Esempio: Si consideri il programma:

P5
```
p(X,f(X)).
q:- p(Y,Y).
```

si ha che la query `?- q.`

ha successo sebbene P5 non derivi q.

Prolog non corretto per la logica delle clausole di Horn.

IL PRINCIPIO DI RISOLUZIONE PER LE CLAUSOLE GENERALI

Clausole nelle quali possono comparire variabili.
Siano C_1 e C_2 due clausole del tipo:

$$C_1 = A_1 \vee ... \vee A_n \qquad C_2 = B_1 \vee ... \vee B_m$$

dove A_i (i=1..n) e B_j (j=1..m) è un letterale positivo o negativo in cui possono comparire variabili.

Se esiste A_i e B_j tali che $[A_i]\theta = [\sim B_j]\theta$, dove θ è la sostituzione unificatrice più generale, allora si può derivare una nuova clausola C_3 (il risolvente):

$$[A_1 \vee .. \vee A_{i-1} \vee A_{i+1} \vee .. \vee A_n \vee B_1 \vee ... \vee B_{j-1} \vee B_{j+1} \vee .. \vee B_m]\theta$$

Date due clausole C_1 e C_2, il loro risolvente C_3 è conseguenza logica di $C_1 \cup C_2$.

Applicazione del mgu per ricavare il risolvente.

Regola di inferenza:

$$\frac{A_1 \vee ... \vee A_n \; B_1 \vee ... \vee B_m \; \exists\theta : [A_i]\theta = [\sim B_j]\theta}{[A_1 \vee ... \vee A_{i-1} \vee Ai +_1 \vee ... \vee A_n \vee B_1 \vee ... \vee B_{j-1} \vee B_{j+1} \vee ... \vee B_m]\theta}$$

dove θ è la sostituzione più generale per A_i e $\sim B_j$.

Esempio

Si considerino le seguenti clausole:

$p(X,m(a)) \vee q(S,a) \vee c(X)$ (C1)

$\sim p(r,Z) \vee c(Z) \vee \sim b(W,U)$ (C2)

I letterali p(X,m(a)) e p(r,Z) sono unificabili attraverso la mgu $\theta=\{X/r, Z/m(a)\}$.

Risolvente:

$q(S,a) \vee c(r) \vee c(m(a)) \vee \sim b(W,U)$ (C3)

Esempio

Dalle seguenti clausole:

$p(a,b) \vee q(X,a) \vee c(X)$ (C1)

$\sim p(a,Y) \vee \sim c(f(Y))$ (C2)

si ottengono i seguenti risolventi:

$q(X,a) \vee c(X) \vee \sim c(f(b))$ (C3)

$p(a,b) \vee q(f(Y),a) \vee \sim p(a,Y)$ (C4)

C3: ottenuto selezionando p(a,b) da C1 e ~p(a,Y) da C2 e applicando la mgu $\theta=\{Y/b\}$.

C4: ottenuto selezionando c(X) da C1 e ~c(f(Y)) da C2 e applicando la mgu $\theta=\{X/f(Y)\}$.

DIMOSTRAZIONE PER CONTRADDIZIONE ATTRAVERSO LA RISOLUZIONE

Dati gli assiomi propri H di una teoria e una formula F, derivando da $H \cup \{\sim F\}$ la contraddizione logica si dimostra che F è un teorema della teoria.

1) Ridurre H e il teorema negato ~F in forma a clausole.

H trasformato nell'insieme di clausole H^C: $H \rightarrow H^C$

F negata e trasformata nell'insieme di clausole F^C:

$\sim F \rightarrow F^C$

2) ***All'insieme*** $\mathbf{H^C \cup F^C}$ ***si applica la risoluzione***

Se F è un teorema della teoria, allora la risoluzione deriva la *contraddizione logica* (clausola vuota) in un numero finito di passi.

Contraddizione: Nella derivazione compariranno due clausole del tipo A e ~B con A e B formule atomiche unificabili.

Per dimostrare F, il metodo originario (Robinson) procede generando i risolventi per ***tutte le coppie*** di clausole dell'insieme di partenza $C_0 = H^C \cup F^C$ che sono aggiunti a C_0. Procedimento iterato, fino a derivare, se è possibile, la clausola vuota.

1. $C_{i+1} = C_i \cup \{\text{risolventi delle clausole di } C_i\}$
2. Se C_{i+1} contiene la clausola vuota, termina. Altrimenti ripeti il passo 1.

Esempio

Per semplicità ci si è limitati al caso di proposizioni:

$H = \{(a \rightarrow c \vee d) \wedge (a \vee d \vee e) \wedge (a \rightarrow \sim c)\}$

$F = \{d \vee e\}$

La trasformazione in clausole di H e ~F produce:

$H^C = \{\sim a \vee c \vee d,\ a \vee d \vee e,\ \sim a \vee \sim c\}$

$F^C = \{\sim d,\ \sim e\}$

Si vuole dimostrare che $H^C \cup F^C$:

$\{\sim a \vee c \vee d,$	(1)	
$a \vee d \vee e,$	(2)	
$\sim a \vee \sim c,$	(3)	
$\sim d,$	(4)	
$\sim e\}$	(5)	

è contraddittorio.

Tutti i possibili risolventi al passo 1 sono:

$\{c \vee d \vee e,$	(6)	da (1) e (2)
$d \vee e \vee \sim c,$	(7)	da (2) e (3)
$\sim a \vee c,$	(8)	da (1) e (4)
$a \vee e,$	(9)	da (2) e (4)
$a \vee d,$	(10)	da (2) e (5)
$\sim a \vee d\}$	(11)	da (1) e (3)

Al passo 2, da (10) e (11) viene derivato il risolvente:

d	(12)

e al passo 3, da (4) e (12) viene derivata anche la clausola vuota.

Esempio

Si considerino gli insiemi:

$H = \{\forall X\ (uomo(X) \rightarrow mortale(X)),\ uomo(socrate)\}$

$F = \{\exists X\ mortale(X)\}$

La trasformazione in clausole di H e ~F produce:

$H^C = \{\sim uomo(X) \vee mortale(X),\ uomo(socrate)\}$

$F^C = \{\sim mortale(X)\}$

L'insieme $H^C \cup F^C$ è il seguente:

$\{\sim uomo(X) \vee mortale(X),$	(1)
$uomo(socrate),$	(2)
$\sim mortale(Y)\}$	(3)

La variabile X di F^C rinominata con Y che non appare in nessuna altra clausola dell'insieme. Questa operazione viene eseguita ogni volta che si aggiungono nuovi risolventi all'insieme delle clausole.

Al passo 1, tutti i possibili risolventi, e le sostituzioni unificatrici più generali applicate per derivarli, sono:

$\{mortale(socrate),$	(4)	$\theta=\{X/socrate\}$
$\sim uomo(Z)\}$	(5)	$\theta=\{X/Y\}$

Al passo 2, da (2) e (5) viene derivata la clausola vuota, applicando la sostituzione {Z/socrate}. La clausola vuota è anche derivabile dalle clausole (3) e (4) applicando la sostituzione {Y/socrate}.

ESEMPIO:

1. cane(fido).
2. ~abbaia(fido).
3. scodinzola(fido).
4. miagola(geo).
5.~scodinzola(X) or ~cane(X) or amichevole(X).
6.~amichevole(X1) or abbaia(X1) or ~spaventato(Y1,X1).
7.~cane(X2) or animale(X2).
8.~miagola(X3) or gatto(X3).

Goal negato: 9.~cane(X4) or ~gatto(Y)or spaventato(Y,X4).
Da 9 e 1: 10. ~gatto(Y) or spaventato(Y,fido).
Da 10. e 6: 11. ~amichevole(fido) or abbaia(fido) or ~gatto(Y).
Da 11. e 2: 12. ~amichevole(fido) or ~gatto(Y).
Da 8. e 12: 13. ~amichevole(fido) or ~miagola(Y).
Da 13. e 4: 14. ~amichevole(fido).
Da 5. e 14: 15. ~scodinzola(fido) or ~cane(fido).
Da 3. e 15: 16. ~cane(fido)
Da 1. e 16. CONTRADDIZIONE !!

CORRETTEZZA e COMPLETEZZA

Si può dimostrare che sotto opportune strategie, la risoluzione è ***corretta e completa***.

Se viene generata la clausola vuota la teoria $H^C \cup F^C$ è insoddisfacibile e se la teoria $H^C \cup F^C$ è insoddisfacibile la derivazione genera la clausola vuota in un numero finito di passi.

Teorema

(Correttezza e completezza della risoluzione)

Un insieme di clausole è insoddisfacibile se e solo se l'algoritmo di risoluzione termina con successo in un numero finito di passi, generando la clausola vuota.

Il metodo di risoluzione procede esaustivamente generando tutti i possibili risolventi ad ogni passo.

STRATEGIE

Si definiscono *strategie* che scelgono opportunamente le clausole da cui derivare un risolvente.

I metodi di prova che si ottengono risultano più efficienti anche se in alcuni casi possono introdurre incompletezza.

La dimostrazione attraverso il principio di risoluzione può essere rappresentata con un grafo, detto ***grafo di refutazione***.

Le clausole dell'insieme base $H^C \cup F^C$ sono nodi del grafo dai quali possono solo uscire archi.

Un risolvente corrisponde a un nodo nel quale entrano almeno due archi (ciascuno da una delle due clausole "parent").

Strategia in ampiezza ("***breadth-first***"). Al passo i (i•0), genera tutti i possibili risolventi a livello i+1-esimo utilizzando come clausole "parent" una clausola di C_i (cioè una clausola a livello i) e una di Cj ($j \leq i$), cioè una clausola appartenente a un livello uguale o minore di i.

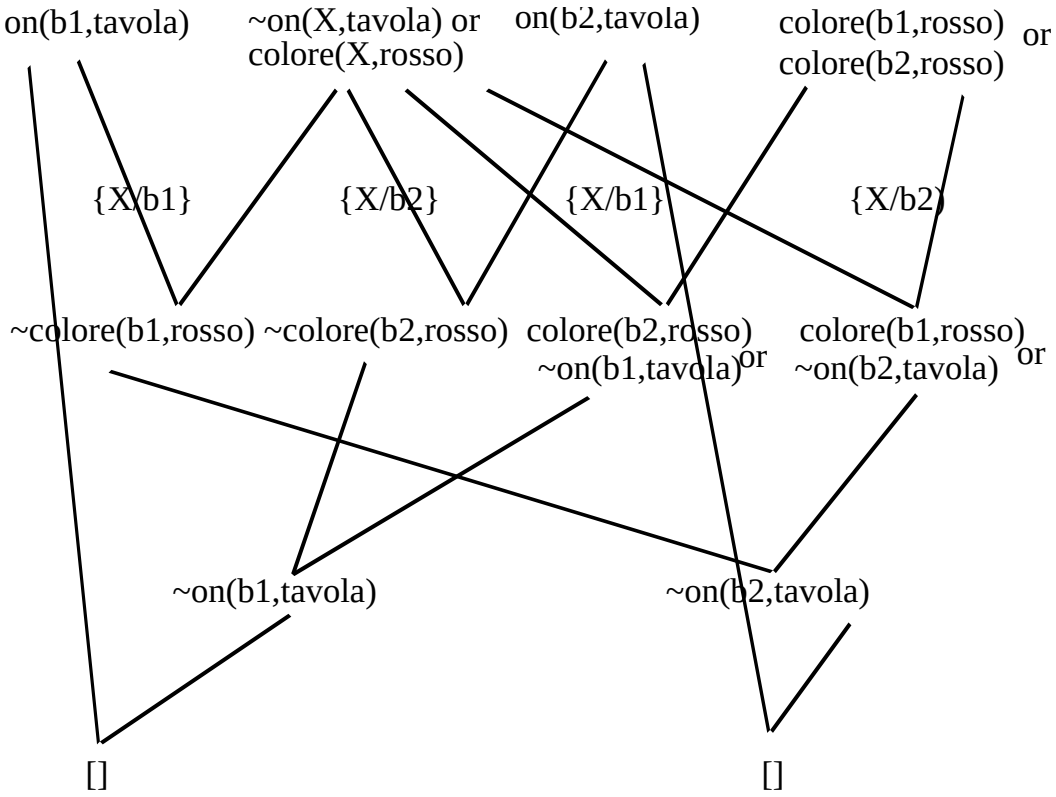
Esempio

$H = H^C$ = {on(b1,tavola), (1)
on(b2,tavola), (2)
colore(b1,rosso) $\vee$ colore(b2,rosso)(3)

F = {$\exists$X (on(X,tavola) $\wedge$ colore(X,rosso))}

F^C = {~on(X,tavola) $\vee$ ~colore(X,rosso)}

L'applicazione della strategia in ampiezza produce il ***grafo di refutazione*** riportato nel seguito:



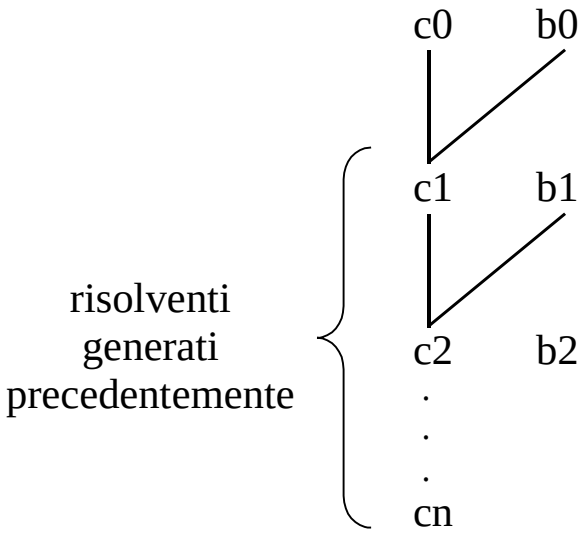
Strategia ***set-of-support*** (completa) sceglie almeno una clausola "parent" fra quelle derivate dalla negazione della formula che si vuole dimostrare o dai loro discendenti.

Strategia ***lineare*** (completa) sceglie almeno una clausola "parent" dall'insieme base C_0 oppure tra i risolventi generati precedentemente. La seconda clausola parent è sempre il risolvente ottenuto al passo precedente.

Nel caso di risoluzione lineare, il grafo di refutazione diventa un albero, detto *albero di refutazione*.

Albero lineare

c_0 appartiene a C_0 (cioè all'insieme base) e b_i appartiene a C_0 oppure è uguale a c_j con $j<i$.



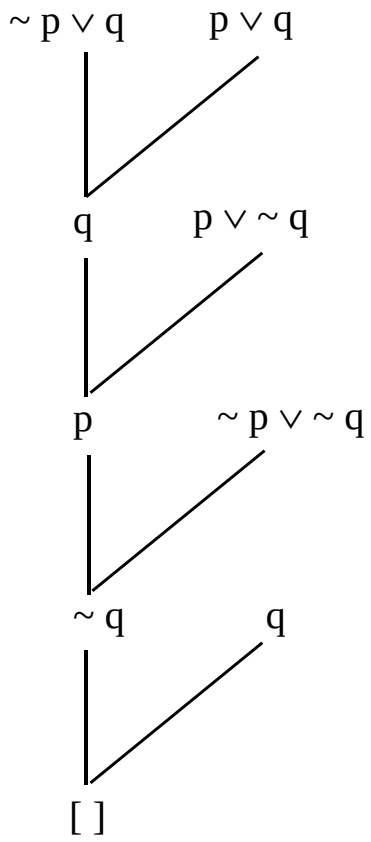
Esempio

C0={p∨q, ~p∨q, p∨~q, ~p∨~q}, ottenuto da:

H = {q→p, ~q→p, p→q}

F = q∧p

Le clausole "parent" rappresentate sulla destra dell'albero sono formule di C0 (le prime tre) e un risolvente generato in precedenza (la formula "q").



Esempio

H^C={sum(0,X1,X1), sum(s(X),Y,s(Z)) ∨ ~sum(X,Y,Z)}

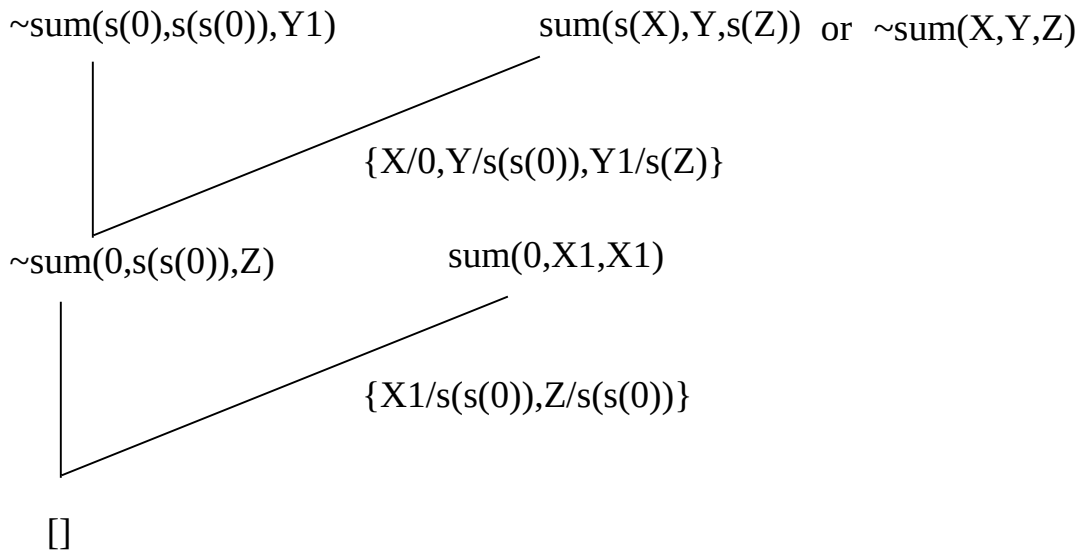
F^C = {~sum(s(0),s(s(0)),Y1)}

dove F^C è stato ottenuto negando la formula:

F = ∃Y sum(s(0),s(s(0)),Y)

L'insieme base $C_0=H^C \cup F^C$ risulta:

C_0 = {sum(0,X1,X1), sum(s(X),Y,s(Z)) ∨ ~sum(X,Y,Z),
~sum(s(0),s(s(0)),Y1)}

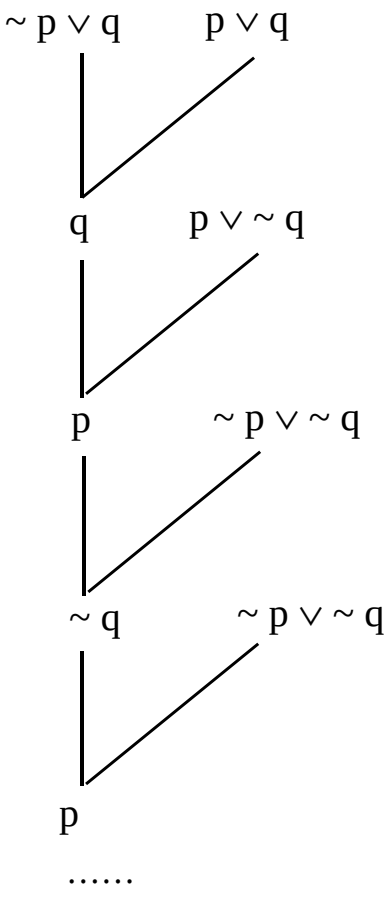


Strategie complete producono comunque grafi di refutazione molto grandi → inefficienti → strategie non complete.

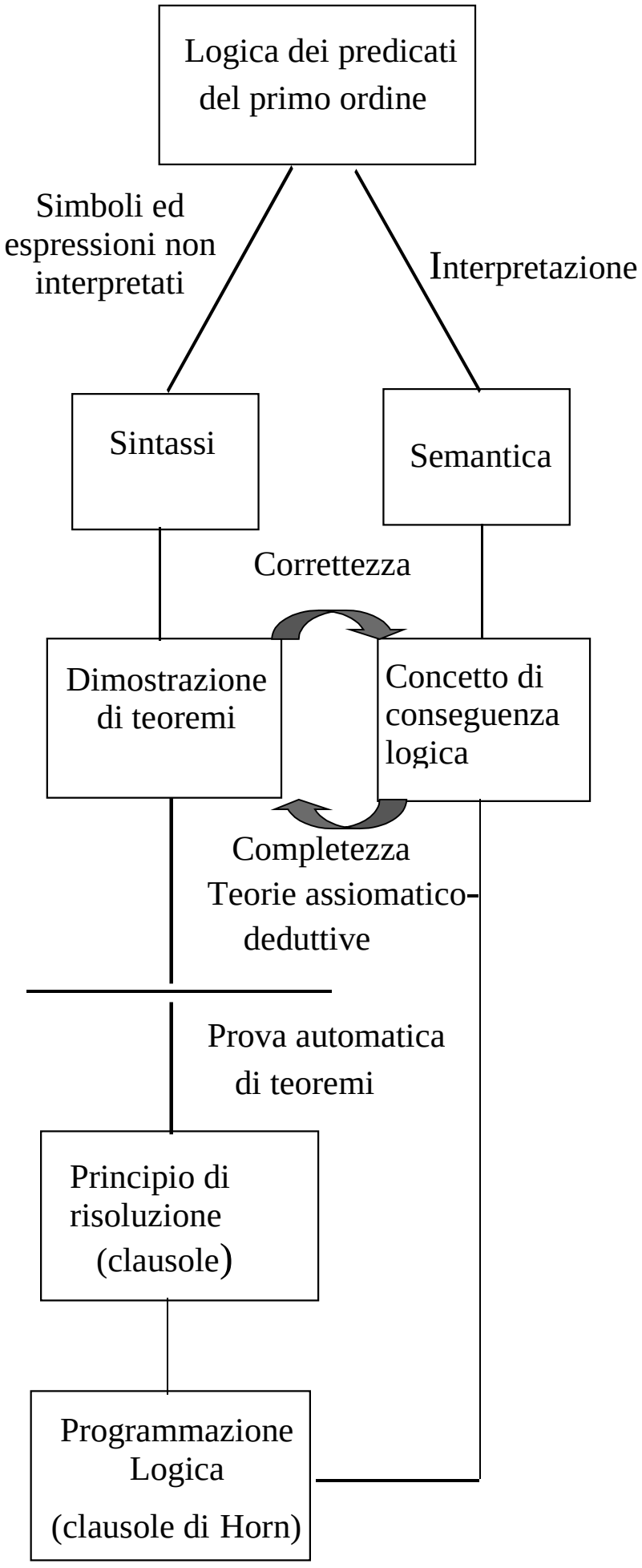
Strategia ***linear-input*** non completa sceglie che ha sempre una clausola “parent” nell'insieme base C_0, mentre la seconda clausola “parent” è il risolvente derivato al passo precedente.
Caso particolare della risoluzione lineare:
vantaggio: memorizzare solo l'ultimo risolvente
svantaggio: non completa

Esempio

C0={p∨q, ~p∨q, p∨~q, ~p∨~q} insoddisfacibile, ma la risoluzione con strategia “linear-input” produce sempre clausole che hanno almeno un letterale, e quindi non è in grado di derivare la clausola vuota.



Se ci si limita, però, a un sottoinsieme delle clausole, in particolare alle clausole di Horn, allora la strategia ***“linear-input” è completa***.


Logica dei predicati del primo ordine
Simboli ed espressioni non interpretati
Interpretazione
Sintassi
Semantica
Correttezza
Dimostrazione di teoremi
Concetto di conseguenza logica
Completezza
Teorie assiomatico-deduttive
Prova automatica di teoremi
Principio di risoluzione (clausole)
Programmazione Logica
(clausole di Horn)

LE CLAUSOLE DI HORN

La logica a clausole di Horn è un sottoinsieme della logica a clausole

Le clausole di Horn hanno al più un letterale positivo.

Le clausole possono essere scritte in una forma equivalente sostituendo al simbolo di disgiunzione $\vee$ e negazione ~ il simbolo di implicazione logica ($\rightarrow$) ricordando che:

$\sim B \vee A$ equivale a $B \rightarrow A$

Nel seguito si indicherà l'implicazione logica $B \rightarrow A$ con:

$A \leftarrow B$ (“B implica A”, oppure “A se B”).

La clausola: $A_1 \vee ... \vee A_n \vee \sim B_1 \vee ... \vee \sim B_m$
può essere riscritta come: $A_1,...,A_n \leftarrow B_1,...,B_m$
dove i simboli “,” che separano gli atomi A_i sono da interpretare come disgiunzioni, mentre quelli che separano gli atomi B_j sono congiunzioni.

Clausole di Horn:

$A \leftarrow B_1,...,B_m$
$\leftarrow B_1,...,B_m$ Goal

Esempio

H={on(b1,tavolo)∧on(b2,tavolo))∧

((colore(b1,rosso)∨colore(b2,rosso)))}

è esprimibile in clausole, ma non in clausole di Horn.

La sua trasformazione in clausole risulterà infatti:
H^C={on(b1,tavolo),on(b2,tavolo),
colore(b1,rosso) ∨colore(b2,rosso)}
dove la clausola:
colore(b1,rosso) ∨ colore(b2,rosso)
non è una clausola di Horn.

Risoluzione per le clausole di Horn: ***risoluzione SLD*** resolution with **S**election rule, **L**inear input strategy for **D**efinite clauses

Risoluzione SLD opera per *contraddizione* e quindi si procede negando la formula F da dimostrare.

Poichè F è una congiunzione di formule atomiche quantificate esistenzialmente, la sua negazione produrrà una disgiunzione di formule atomiche negate quantificata universalmente, cioè una clausola di Horn goal.

Esempio

La negazione della formula:

$F=\exists W\ sum(s(0),0,W)$

è la clausola goal:

$\leftarrow sum(s(0),0,W).$

Le negazioni delle formule:

$\exists Y\ sum(s(0),s(s(0)),Y)$

$\exists Y\ \exists Z\ sum(s(0),s(s(0)),Y) \wedge sum(Y,s(0),Z)$

sono rispettivamente:

$\leftarrow sum(s(0),s(s(0)),Y)$

$\leftarrow sum(s(0),s(s(0)),Y),sum(Y,s(0),Z).$