Paradigmi e Linguaggi di Programmazione

Programmazione Logica

Logica

- La logica è tradizionalmente studiata in linguistica, filosofia e matematica
- Rappresenta la via dell'argomentazione e della dimostrazione
- Già gli antichi Greci distinguevano tra
  - enunciati: affermazioni pure e semplici
  - teoremi: affermazioni dotate di una dimostrazione

- La necessità pratica di distinguere tra enunciati corretti ed errati ha portato alla formalizzazione delle dimostrazioni.

Logica

- La logica matematica nasce con Boole e con la sua idea di *quantificare i predicati*, cioè di applicare alla vecchia logica formale di derivazione aristotelica le regole e i procedimenti dell'algebra
  - ogni ragionamento poteva venir ridotto ad un puro calcolo formale.

Logica

- La logica formale si occupa della *formalizzazione* del linguaggio naturale (affetto da ambiguità e ridondanze) e della costruzione di calcoli capaci di garantire ragionamenti rigorosi e non intuitivi.
  - Dato un frammento di linguaggio naturale, lo si analizza individuando le componenti che hanno rilevanza logica.

- Il grado di profondità dell'analisi determina la complessità del linguaggio artificiale della logica.

Logica

- La logica proposizionale e la logica dei predicati costituiscono due diversi livelli di approfondimento:
  - entrambe hanno come oggetto l'analisi di frasi dichiarative, quelle per cui ha senso chiedere se siano vere o false.
  - La *logica proposizionale* considera come unità base dell'analisi le proposizioni.
  - La *logica dei predicati* del primo ordine approfondisce l'analisi delle proposizioni in termini di soggetto-predicato.

Logica proposizionale (linguaggio formale)

Il linguaggio della logica proposizionale è formato da un alfabeto e da una grammatica (che ne costituiscono la **sintassi**).

- L'**alfabeto** è costituito da:
  - Un insieme L infinito numerabile di lettere proposizionali.
  - Un insieme di simboli per i connettivi vero-funzionali (not, &, v, ==>, <==>).
  - Un insieme di simboli ausiliari: parentesi.

Logica proposizionale (linguaggio formale)

- La **grammatica** della logica proposizionale sono regole ricorsive che permettono di definire proposizioni, cioè formule di complessità qualunque:
  - ogni lettera proposizionale è una formula atomica;
  - ogni formula atomica è una formula;
  - se X è una formula lo è anche notX;
  - se X e Y sono formule allora lo sono anche:
  - (X&Y), (XvY), (X==>Y), (X<==>Y);
- La **Semantica** sul concetto di valutazione (attraverso l'uso delle tavole di verità)

Logica proposizionale

Proprietà semantiche definite mediante «valutazione»

- Verità: Una formula X è vera se e solo se una valutazione v la rende vera.
- Validità :Una formula P è valida se e solo se ogni valutazione v la rende vera. Formule di questo tipo sono dette tautologie.
- Falsità: Una formula P è falsa se e solo se ogni valutazione v la rende falsa.
- Inconsistenza: Un insieme di formule si dice inconsistente se e solo se non è consistente.
- Equivalenza: Due proposizioni P e Q sono equivalenti se e solo se per ogni valutazione v, P e Q assumono lo stesso valore.

Logica proposizionale

Proprietà semantiche definite mediante "valutazione"

- Consistenza : Un insieme di formule è consistente se e solo se c'è almeno una valutazione per cui tutti gli elementi dell'insieme sono veri.
- Implicazione logica :Un insieme F di formule implica logicamente una formula P se e solo se non c'è alcuna valutazione per cui tutti gli elementi di F sono veri e P è falsa. Esiste un simbolo per l'implicazione logica: ==>

E' possibile dimostrare le proprietà semantiche possono essere definite a partire dal ***concetto di inconsistenza****: mostrando che la negazione di un insieme di formule è inconsistente si prova la sua validità.*

Logica del primo ordine

- Si estende la possibilità di formalizzare il linguaggio naturale approfondendo l'analisi delle proposizioni.
- Il linguaggio della logica proposizionale viene arricchito di tre nuovi elementi:
  - lettere predicative,
  - termini,
  - quantificatori.

Logica del primo ordine

- Con questo alfabeto allargato siamo in grado di formalizzare frasi del tipo "*Ogni uomo ha una madre" tenendo conto della sua struttura interna*.
- Infatti, parafrasando, otteniamo "*per ogni individuo x che sia uomo esiste un individuo y tale che x è madre di y*". In formula questo viene tradotto così:

x [U(x) ==> Vy(M(y,x)]

- dove U simboleggia il predicato "essere uomo" e M il predicato "essere madre".

Logica del primo ordine

Sintassi

- ***Alfabeto***
  - lettere predicative (A, B, C, ... , Z).
  - costanti individuali (a, b, c, ...).
  - variabili individuali (x, y, z, ...).
  - lettere funzionali (f(g, h, ...).
  - connettivi logici (not, &, v, ==>, <==>).
  - simboli per quantificatori (quantificatore universale V, quantificatore esistenziale E).
  - simboli ausiliari (virgole e parentesi).

- Definizione di ***espressione***
  - Chiamiamo espressione una qualsiasi successione finita di simboli dell'alfabeto.

Logica del primo ordine

Grammatica

- si dice formula atomica A(t1,...,tn) una formula ottenuta per applicazione di una lettera predicativa (A) a termini (t1,....,tn).se P è una formula, allora lo è anche notP.
- se P e Q sono formule, lo sono anche (P&Q), (PvQ), (P==>Q), (P<==>Q).
- se P è una formula e x una variabile individuale, allora (Vx)P e (Ex)P sono formule.
- nient'altro è una formula.

Logica del primo ordine

Semantica

Il concetto semantico base è quello di "***interpretazione***".

- Una interpretazione interpreta ogni costante individuale, ogni lettera predicativa, ogni lettera funzionale relativamente ad un "universo del discorso" (D) che è un insieme non vuoto di elementi.

- Esempio:
  - *D = l'insieme delle creature viventi*.
  - F(x) = x è umano.

Logica del primo ordine

Definizione di ***interpretazione***

- Una interpretazione di una formula P di F è costituita da un dominio non vuoto D e da una assegnazione di "valori" ad ogni costante, simbolo di funzione e simbolo di predicato presente in P nel modo seguente:
  - ad ogni costante si assegna un elemento di D;
  - ad ogni simbolo di funzione n-aria si assegna una funzione n-aria del dominio D;
  - ad ogni simbolo di predicato n-ario si assegna un insieme di n-uple di elementi di D.

Logica del primo ordine

Proprietà semantiche

- Validità: Un enunciato P è valido <==> è vero in ogni interpretazione.
- Falsità: Un enunciato P è falso <==> è falso in ogni interpretazione.
- Inconsistenza: Un insieme di enunciati è inconsistente <==> l'insieme non è consistente.
- Equivalenza: Due enunciati P e Q sono equivalenti <==> in nessuna interpretazione P e Q hanno valori differenti.

Logica del primo ordine

Proprietà semantiche

- Consistenza: Un insieme di enunciati è consistente <==> c'è almeno una interpretazione in cui tutti gli elementi dell'insieme sono veri.
- Implicazione logica: Un insieme G di enunciati implica logicamente un enunciato P <==> in nessuna interpretazione tutti gli elementi di G sono veri e P è falso.

Come nella logica proposizionale è possibile dimostrare le proprietà semantiche possono essere definite a partire dal ***concetto di inconsistenza****: mostrando che la negazione di un insieme di formule è inconsistente si prova la sua validità.*

Forma a clausole

- E’ un formalismo che, al pari della forma standard della logica, permette la formalizzazione del linguaggio naturale e di conseguenza la possibilità di costruire calcoli rigorosi al posto dei ragionamenti informali propri del linguaggio naturale.

- È però un formalismo più compatto e semplice rispetto alla forma standard.

- Si dimostra che sia S un insieme di clausole che rappresenta una forma standard di una formula F se F è inconsistente <==> S è inconsistente

Forma a clausole

- Il considerare la forma clausale è un passo obbligato nel cammino che dalla logica matematica conduce alla programmazione logica.

- Infatti, sia il formalismo che la regola di inferenza che viene qui applicata (il Principio di Risoluzione) sono molto più simili ai convenzionali linguaggi di programmazione che il formalismo e le regole di inferenza della forma standard della logica matematica.

Forma a clausole e Clausole di Horn

- In logica, una **clausola** è una disgiunzione logica fra formule atomiche: A ∨ B ∨ ¬C

- La ***clausola di Horn*** è una disgiunzione di letterali in cui al massimo un letterale è positivo :
  - ¬a ∨ b ∨ c,
  - ¬a ∨ ¬b ∨ ¬c

Le clausole di Horn sono alla base della programmazione logica ma anche dei vincoli di integrità di un database.

Forma a clausole e Clausole di Horn

- Le clausole di Horn possono essere rappresentate sotto forma di implicazione logica.
  - Dati i letterali negativi (a, b), detta "premessa" o "corpo", si implica come conseguenza logica il terzo letterale (c), detto "conclusione" o "testa", che può essere positivo o negativo.
  - $(a \wedge b) \Rightarrow c$
- Una clausola di Horn senza letterali positivi può essere rappresentata come un'implicazione la cui conclusione è falsa.
  - $(a \wedge b) \Rightarrow \neg c$

Clausole di Horn e Principio di Risoluzione

- Il **Principio di Risoluzione**, introdotto da Robinson nel 1965, è una regola d'inferenza applicabile a tutti gli insiemi di clausole, semplice ed estremamente potente, facile da analizzare e da implementare.

- Il Principio di Risoluzione si usa per verificare l'insoddisfacibilità di un insieme S di clausole.
- L'idea è quella di controllare se S contiene la clausola vuota (che corrisponde ad una contraddizione): in questo caso risulta insoddisfacibile.

- Se S non contiene la clausola vuota, si controlla se questa può essere derivata da S.

Clausole di Horn e Principio di Risoluzione

Definizione informale del principio di risoluzione

- Per ogni coppia di clausole C1 e C2 (dette clausole genitrici), se esiste un letterale L1 in C1 che risulti complementare ad un letterale L2 in C2, si cancellino L1 ed L2, rispettivamente da C1 e da C2, e si costruisca la disgiunzione delle clausole ottenute.
- La clausola così costruita è detta **risolvente** di C1 e C2.

- Esempio:
  - C1: P(x), ¬R(a)
  - C2: S(y), R(a)
  - C, Risolvente di C1 e C2: P(x), S(y).

Clausole di Horn e Principio di Risoluzione

Alberi di deduzione (o refutazione)

- I processi di refutazione che usano la risoluzione si visualizzano con strutture a grafo (gli alberi di deduzione) in cui ogni nodo è una clausola.
- Dato un insieme di clausole S, le clausole sono le foglie e se due clausole corrispondenti risolvono, si ha una risolvente detta discendente immediata delle due clausole.
- ***La radice di un grafo di refutazione con risoluzione è la clausola vuota.***

Clausole di Horn e Principio di Risoluzione

Esempio

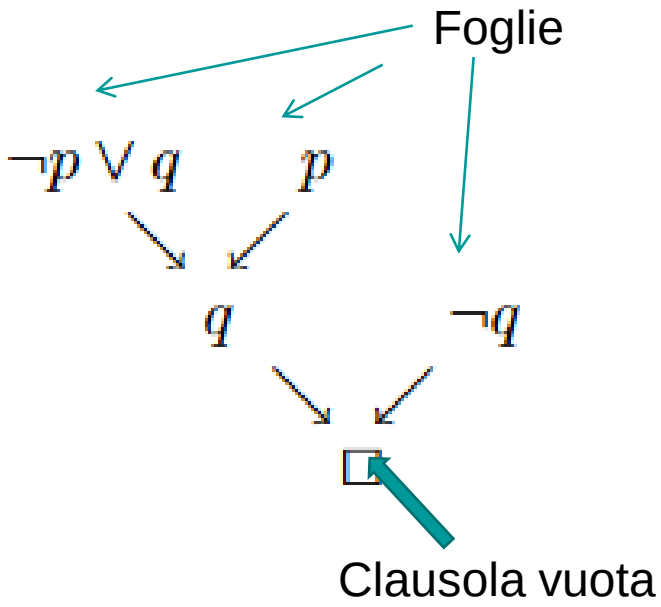
Dato l'insieme di clausole:

S={¬pvq, p,¬q}

l'albero di deduzione è una refutazione di S

(trovo la clausola vuota

quindi S è inconsistente)



Clausole di Horn e Principio di Risoluzione

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).

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

Da 1. e 16. trovo la clausola vuota (CONTRADDIZIONE !!)

Clausole di Horn e Principio di Risoluzione

- Si consideri la seguente conoscenza:
  - Antonio, Michele e Giovanni sono iscritti al CAI (Club Alpino Italiano). Ogni appartenente al Club che non è sciatore è uno scalatore. Gli scalatori non amano la pioggia. Ogni persona che non ama la neve non è uno sciatore. Antonio non ama ciò che Michele ama. Antonio ama la pioggia e la neve.

- Si rappresenti tale conoscenza come un insieme di predicati del primo ordine appropriati per un sistema di refutazione che lavori mediante risoluzione.

- Si mostri come tale sistema risolverebbe la domanda: "C'è un membro del CAI che è uno scalatore, ma non uno sciatore?

Clausole di Horn e Principio di Risoluzione

- Formule logiche della base di conoscenza:
  - 1.∀X iscritto(X), not sciatore(X)→scalatore(X)
  - 2.∀X scalatore (X)→not ama(X,pioggia)
  - 3.∀X not ama(X,neve)→not sciatore(X)
  - 4.∀X ama(michele,X)→not ama(antonio,X)
  - 5. ama(antonio,neve)
  - 6. ama(antonio, pioggia)
  - 7. iscritto(antonio)
  - 8. iscritto(michele)
  - 9. iscritto(giovanni)
- Goal:
  - ∃X iscritto(X), scalatore(X), not sciatore(X)

Clausole di Horn e Principio di Risoluzione

- Forma a clausole:
  - C1. not iscritto(X)vsciatore(X)vscalatore(X)
  - C2. not scalatore(X)vnot ama(X,pioggia)
  - C3. ama(X,neve)vnot sciatore(X)
  - C4. not ama(michele,X)vnot ama(antonio,X)
  - C5. ama(antonio,neve)
  - C6. ama(antonio, pioggia)
  - C7. iscritto(antonio)
  - C8. iscritto(michele)
  - C9. iscritto(giovanni)
- Gneg:
  - not iscritto(X)vnot scalatore(X)vsciatore(X)

Clausole di Horn e Principio di Risoluzione

- Risoluzione
  - C10=Gneg - C8 => not scalatore(michele) v sciatore(michele) {X/michele}
  - C11=C10 - C3 => not scalatore(michele) v ama(michele,neve)
  - C12=C11- C4 => not scalatore(michele) v not ama(antonio,neve)
  - C13=C12 e C5 => not scalatore(michele)
  - C14=C13 e C1 not iscritto(michele) v sciatore(michele)
  - C15=C13 e C8 => sciatore(michele)
  - C16=C15 e C3 => ama(michele,neve)
  - C17=C16 e C4 => not ama(antonio,neve)
  - C18=C17 e C5 => clausola vuota

Programmazione Logica

- E’ un paradigma di programmazione che adotta la logica matematica sia per rappresentare sia per elaborare l'informazione:
  - un problema viene descritto con un insieme di formule della logica, dunque in forma dichiarativa.
- L'interpretazione procedurale delle clausole consente di usare alcune tecniche di dimostrazione di teoremi come meccanismi di esecuzione dei programmi.
  - logica = rappresentazione del problema
  - deduzione = risoluzione del problema

Programmazione Logica

- La programmazione logica differisce dalla programmazione tradizionale, in quanto richiede e nello stesso tempo consente al programmatore di descrivere la struttura logica del problema piuttosto che il modo di risolverlo.

- Il programmatore si può così concentrare sugli aspetti logici del problema e sul modo migliore per rappresentarli, senza essere focalizzato sulla necessità di determinare in dettaglio il modo di pervenire ai risultati.

Programmazione Logica

- La programmazione classica, procedurale, è più adatta a problemi quotidiani, ben definiti.
- Essa adotta un **paradigma imperativo**: richiede, cioè, che siano specificate delle rigorose sequenze di passi (algoritmi) che, a partire dai dati a disposizione, portino ad ottenere i risultati desiderati.
- Nei programmi classici i dati (cosa) e il controllo (come) si intersecano tra loro formando un agglomerato inscindibile.
  - Algoritmi + Strutture dati = Programmi
- Vengono generalmente lasciate implicite le assunzioni su cui l'algoritmo si basa. (stile prescrittivo, how-type).

Programmazione Logica

- Nella programmazione logica il programmatore dovrebbe preoccuparsi di specificare solo la componente logica, mentre il controllo dovrebbe essere appannaggio esclusivo del sistema.
- Cioè deve solo definire il problema, formulandone le specifiche, senza dire alla macchina come risolverlo.
- Algoritmo = Logica + Controllo
- *Poiché il controllo è demandato totalmente al sistema, eventuali modifiche possono solo influenzare l'efficienza, non la correttezza dei risultati.*

- I programmi si scrivono descrivendo la conoscenza relativa al problema, cioè specificando gli oggetti che vi intervengono e le relazioni fra di essi. (stile descrittivo, what-type)

Programmazione Logica

- Esempio:

Problema: Prendere l'ascensore.

- Linguaggio imperativo
  - Attendere che l'ascensore arrivi al piano di chiamata
  - Aprire la porta dell'ascensore
  - Entrare nell'ascensore
  - Chiudere la porta dell'ascensore
  - Spingere il tasto corrispondente al piano da raggiungere

- Ai fini del risultato, conta l'ordine in cui sono elencate le azioni da intraprendere.

Programmazione Logica

- Esempio:

Problema: Prendere l'ascensore.

- Linguaggio dichiarativo
  - Se l'ascensore è arrivato e la porta è aperta, allora si può entrare nell'ascensore
  - Se si vuole aprire la porta dell'ascensore, bisogna aspettare che l'ascensore arrivi
  - Se si è entrati e la porta è aperta, allora la si può chiudere
  - Se si è entrati e la porta è chiusa, allora si deve spingere il tasto corrispondente al piano da raggiungere

- L'ordine in cui le regole sono elencate non cambia il risultato.

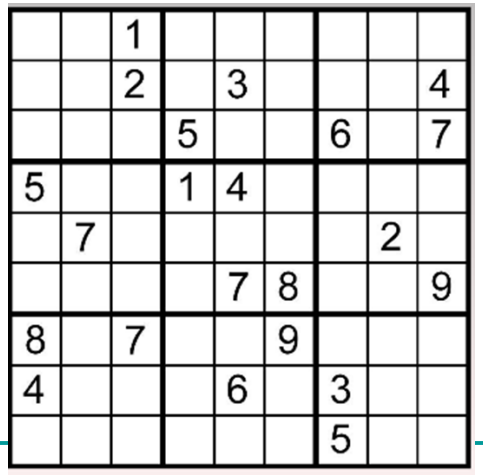
Programmazione Logica

- Nella programmazione logica si deve abbandonare il modo di pensare orientato al processo.
- Il programma è una descrizione della soluzione, non del processo, e si costruisce descrivendo in un linguaggio formale l'area applicativa, ossia gli oggetti che in essa esistono, le relazioni fra loro e i fatti che li riguardano.
  - Gli oggetti possono essere concreti o astratti, esistenti o immaginari.
  - Le relazioni sono qualità o attributi di un gruppo di oggetti, che legano gli uni agli altri.
  - I fatti e le regole asseriti costituiscono le assunzioni del programma (ossia la base di conoscenza del sistema).
- I programmi si mettono in funzione ponendo al sistema delle domande circa gli oggetti del dominio e le loro relazioni.

Sudoku

Problema a vincoli:

- le variabili sono le 81 caselle della griglia,
- il dominio di ogni variabile sono i numeri da 1 a 9
- i vincoli sono la non ripetizione delle cifre in righe, colonne, sottogriglie.
- In questo caso è dato anche un assegnamento iniziale (i numeri già presenti nella griglia) che rende unica la soluzione allo schema.



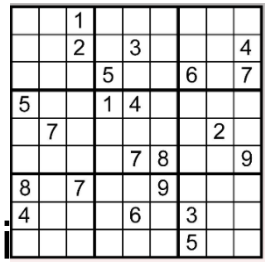
Sudoku

- domain($X_{1,1}$,...,$X_{9,9}$) = [1..9]
- $X_{1,3}$=1,$X_{2,3}$=2,$X_{2,5}$=3,$X_{2,9}$=4,...,$X_{9,7}$=5
- alldifferent($X_{1,1}$,...,$X_{1,9}$)...alldifferent($X_{9,1}$,...,$X_{9,9}$)
- alldifferent($X_{1,1}$,..., $X_{9,1)}$...alldifferent($X_{1,9}$,...,$X_{9,9}$)
- alldifferent($X_{1,1}$,...,$X_{3,3}$)...alldifferent($X_{7,7}$,...,$X_{9,9}$)

		1						
		2		3				4
			5			6		7
5			1	4				
	7						2	
				7	8			9
8		7			9			
4				6		3		
						5		

- Iniziando a controllare che le regole siano verificate all'inserimento dei numeri noti si vede che i numeri che potenzialmente possono essere inseriti nelle altre celle diminuiranno
- Quando un solo numero può comparire in una cella, quello è il numero corretto

Sudoku



- Partendo da un problema Sudoku, si applicano ripetutamente alcune tecniche di riduzione dell'insieme dei numeri ammissibili in ciascuna cella non assegnata (metodi o regole di riduzione).
- Le regole di riduzione non fanno altro che verificare delle condizioni necessarie per una soluzione, non sufficienti.
  - E' per questo motivo che l'applicazione di una o più regole non garantisce di ottenere una soluzione.
  - Una condizione necessaria è del tipo "se x è una soluzione, allora è vera la condizione y"; viceversa una condizione sufficiente sarebbe del tipo "se è vera la condizione w, allora x è una soluzione".
  - Se riuscissimo a trovare una condizione w sufficiente, che si rivelasse vera, riusciremmo a risolvere il Sudoku in un solo passo