SCUOLA INTERATENEO DI
SPECIALIZZAZIONE PER LA FORMAZIONE
DEGLI INSEGNANTI DELLA SCUOLA
SECONDARIA
SIS

Logica e computer: una traccia per una unità didattica

A.A. 2001/2002

Giovanni Nicco

La logica ed il linguaggio Prolog:

In questo lavoro mi propongo di dare alcune indicazioni a un insegnante del biennio di liceo scientifico che intendesse utilizzare il linguaggio PROLOG per condurre una serie di lezioni sulla logica.

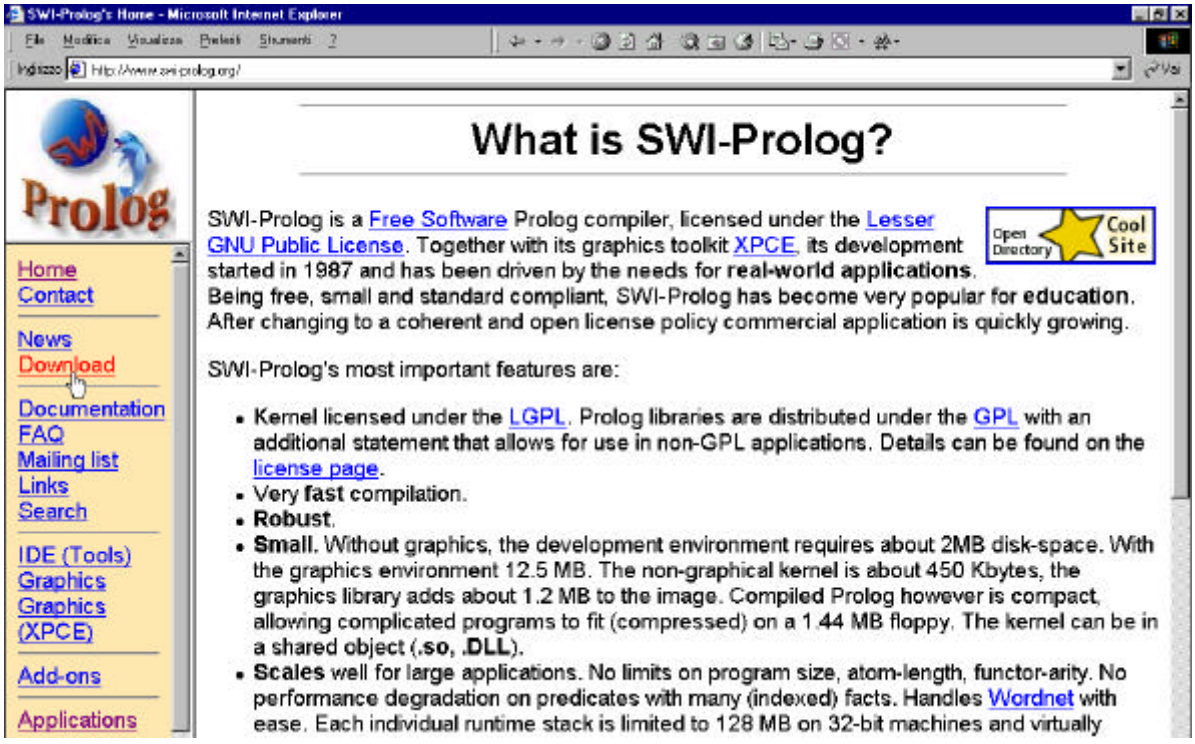
Innanzitutto è necessario procurarsi un interprete PROLOG, un tale programma, gratuito e di facile utilizzo è reperibile al seguente indirizzo internet.

http://www.swi.psy.uva.nl/cgi-bin//nph-download/SWI-Prolog/w32pl328.exe

	963164 Bytes	Last pre-ISO version for MS-Windows Self-installing executable of version **3.2.8** for Windows 95, 98 and Windows-NT

Per una maggiore documentazione è possibile visitare l'intero sito:

http://www.swi-prolog.org/



Il programma si presenta con una interfaccia molto semplice ma ha un piccolo help e il recover dei comandi ne rende agevole l'utilizzo:

Prima di passare alla traccia è opportuno rivedere almeno qualche nozione di logica:

La logica studia le regole che governano la deduzione di frasi vere (teoremi) a partire da altre frasi assunte come vere (assiomi).

Logica delle proposizioni: si occupa di stabilire se una *proposizione* è vera, intendendo per *proposizione* una frase composta da *costanti proposizionali* (a,b,c) e da cinque *connettivi logici*: negazione ¬, congiunzione &, disgiunzione |, implicazione -->, equivalenza <-->

essa ci dice ad esempio che, date le proposizioni
a = Fido è un cane
b= Fido ha la coda
La proposizione
a | b è vera

Una *formula* (in breve una proposizione formata con proposizioni semplici unite da connettivi) si definisce *consistente* o *soddisfacibile* se per determinati valori delle proposizioni componenti essa è vera (es p|q) **(**∃ ***vera)****;*
non valida se se per determinati valori delle proposizioni componenti essa è falsa (es p|q) **(**∃ ***falsa)***;
valida se ogni sua intepretazione (assegnazione di valore alle proposizioni costituenti) la rende vera (tautologia) **(**∀ ***vera)***;
contingente quando esiste almeno una interpretazione che la rende vera ma non tutte (cioè e consistente e non valida) **(**∃ ***vera)& ¬ (***∀ ***vera)***; si dice infine che è *contradditoria* se è sempre falsa**(**∀ ***falsa)***.

Indicando con c la consistenza di una proposizione e con nv la sua non validità si può osservare che:
c(p) : vera
c(p&q): vera se c(p) & c(q)
c(p|q):vera se c(p)|c(q)
c(¬p): nv(p)

nv(p):falsa
nv(p&q):nv(p)|nv(q)
nv(p|q): nv(p)&nv(q)
nv(¬p):c(p)

In PROLOG è possibile scrivere questa constatazione come una *regola* sostituendo alle specifiche proposizioni (p,q) due generiche proposizioni (P,Q):

```
c(true).
c(P and Q):-  c(P) , c(Q).
c(P|Q):- c(P);c(Q).
c(¬P):- nv(P).

nv(false).
nv(P and Q):-nv(P);nv(Q).
nv(P|Q):- nv(P),nv(Q).
nv(¬P):-c(P).
```

Logica dei predicati : si occupa di stabilire il valore di verità di una proposizione contenente oltre a

proposizioni (p,q,r) compatibili con quanto visto per la logica delle proposizioni anche delle

variabili (x,y,z) cioè simboli che stanno per qualcosa che può assumere diversi valori, delle

costanti funzionali (f,g,h) che sono proposizioni quali quelle che si trattano nella logica delle

proposizioni ma che contengono delle variabili , dei

predicati(P,Q,R) che sono delle "funzioni logiche" in cui compare una/più variabili (arità del predicato)

e che sono veri o falsi a seconda di come è stato definito il predicato,

il *quantificatore esistenziale* $\exists$ e il

quantificatore universale $\forall$.

essa ci dice che date due generiche proposizioni P e Q la seguente proposizione è vera:
P|Q <--> (¬P) & (¬Q) (tautologia)
P & (¬P) viceversa è sempre falsa (contraddizione)

In PROLOG i quantificatori sono caratterizzati dalla comparsa, a destra di una regola, di una variabile (lo si denota con una maiuscola) come secondo termine dei predicati o come termine che compare a destra in una regola ma non a sinistra:

antenato(adamo,X).
significa "qualsiasi sia X adamo era un suo antenato"

nonno(X,Y):-genitore(X,Z),genitore(Z,Y).
significa: per ogni X e per ogni Y (X è nonno di Y se esiste Z tale che X è genitore di Z e Z è genitore di Y.

Per una analisi approfondita dell'algoritmo di riduzione/unificazione (basato sul principio di risoluzione) attraverso il quale l'interprete PROLOG analizza tutte le possibili interpretazioni di un *goal* restituendo *false* in caso di insuccesso o viceversa i set di valori delle variabili che hanno permesso di soddisfarlo è possibile fare riferimento al testo "From standard logic to logic programming" di André Thayse. Ed. John Wiley & Sons.

Viceversa è possibile integrare la conoscenza del linguaggio con il testo "PROLOG, linguaggio e applicazioni" Ed. Jackson e trovare esercizi e spunti per le lezioni su "Algebra Informatica Geometria" di P.Oriolo e A.Coda, Ed. Bruno Mondadori.

LOGICA DELLE PROPOSIZIONI

Obiettivi:

- Capire il concetto di proposizione e quello di valore di verità
- Definire i connettivi logici 'non' , 'e', 'o' e saperli utilizzare
- Definire l'equivalenza logica e conoscere le proprietà dei connettivi
- Comprendere il concetto di tautologia e quello di contraddizione

PROPOSIZIONI E LORO VALORE DI VERITA'

Una proposizione è una frase che può essere vera o falsa, è possibile *asserire* che una proposizione è vera dicendolo al programma in PROLOG nel seguente modo:

?- assert('il 25 dicembre è Natale').

lui risponde

Yes

(le scritte in **bold** sono riferite alle risposte del programma)

a questo punto è possibile *chiedere* se la asserzione 'il 25 dicembre è Natale' è vera o falsa:
se risponde **Yes** vuol dire che è vera
se risponde **No** vuol dire che o è falsa o comunque non ha modo di decidere se è vera, proviamo (basta scrivere la proposizione è il programma intepreta che sia una domanda):

?- 'il 25 dicembre è Natale'.

Yes

proviamo viceversa un'altra frase

?- 'oggi è una bella giornata'.

[WARNING: Undefined predicate: `oggi è una bella giornata/0']

No

il programma risponde NO e ci avvisa che non gli è stato mai comunicato nulla sulla proposizione 'oggi è una bella giornata'.
Proviamo a inserire altre 3 asserzioni (cioè proposizioni che affermiamo essere vere):
?- assert('la Sicilia è una isola').
Yes
?- assert('il numero 7 è divisibile per 2').
Yes
?- assert('Milano è una città del Piemonte').
Yes

chiediamo a questo punto se 'Milano è una città del Piemonte' è una proposizione vera:

?- 'Milano è una città del Piemonte'.

Yes

Il programma ci risponde di si anche se noi sappiamo che non è vero: la logica "funziona" solo se le asserzioni di base sono corrette altrimenti dà delle risposte coerenti con le asserzioni di base (postulati) date.

CONNETTIVI LOGICI

considerate le proposizioni (vere)
a: piove
b: 'il mare è calmo'

analizziamo il valore di verità delle seguenti proposizioni composte

1) piove **e** il mare è calmo
2) **non** piove **e** il mare è calmo
3) piove **e** il mare non è calmo (cioè: piove **e non**(il mare è calmo))
4) piove **o** il mare è calmo

la **e** si scrive con la ","
la **o** con il ";"
il **non** con il "not"

1)

?- assert(piove).
Yes
?- assert('il mare è calmo').
Yes
?- piove,'il mare è calmo'.
Yes

2)

?- not piove, 'il mare è calmo'.
No

3)

?- piove, not 'il mare è calmo'.
No

4)

?- piove;'il mare è calmo'.
Yes

Studiamo in dettaglio i connettivi logici ricordando che in PROLOG "false" esiste solo come "not true"

e (and)

?- not true,not true.
No
?- true,not true.
No
?- not true,true.
No
?- true,true.
Yes

o (or)

?- not true;not true.
No
?- true; not true.
Yes
?- not true;true.
Yes
?- true;true.
Yes

non (not)

?- not(not true).
Yes
?- not(true).
No

TAVOLE DELLA VERITA', PROPOSIZIONI EQUIVALENTI E PROPRIETA'

Costruiamo la tavola della verità di P&(Q|R) e vediamo se è la stessa di (P&Q)|(P&R), se si diciamo che le due proposizioni composte sono equivalenti.

```
?- assert(f1(P,Q,R):-P,(Q;R)).
?- assert(f2(P,Q,R):-(P,Q);(P,R)).
```

```
?- f1(not true,not true,not true).
No
?- f2(not true,not true,not true).
No

?- f1(not true,not true, true).
No
?- f2(not true,not true, true).
No

?- f1(not true,true,not true).
No
?- f2(not true,true,not true).
No

?- f1(not true,true, true).
No
?- f2(not true,true, true).
No

?- f1( true,not true,not true).
Yes
?- f2( true,not true,not true).
No

?- f1( true,not true, true).
Yes
?- f2( true,not true, true).
Yes

?- f1( true, true,not true).
Yes
?- f2( true, true,not true).
Yes

?- f1( true, true, true).
Yes
?- f2( true, true, true).
Yes
```

Ogni gruppo può dimostrare una equivalenza diversa per raccogliere le proprietà dei connettivi logici:
P&Q=Q&P (commutativa di &) ; P&P=P (idempotenza); P|Q=Q|P (commutativa di |)
(P&Q)&R=P&(Q&R) (associativa di &); (P|Q)|R=P|(Q|R) (associativa di |)
P&(Q|R)=(P&Q)|(P&R) (distributiva di & rispetto a |)
P|(Q&R)=(P|Q)&(P|R) (distributiva di | rispetto a &)
Tautologia: verificare che P| ¬P è sempre vera qualsiasi sia il valore di P
Contraddizione: verificare che P|&¬P è sempre falsa qualsiasi sia il valore di P
Prima legge di De Morgan: verificare che ¬(P&Q)= (¬P|¬Q)
Seconda legge di De Morgan: verificare che ¬(P|Q)= (¬P&¬Q)

LOGICA DEI PREDICATI

Obiettivi:

- Capire il concetto di argomento e quello di predicato
- Capire il concetto di proposizione aperta
- Sapere individuare l'insieme universo di un predicato e, al suo interno, l'insieme di verità
- Conoscere e sapere utilizzare i quantificatori
- Comprendere l'importanza della correlazione tra le operazioni con gli insiemi e le operazioni con le proposizioni

Data la proprietà *pari* solo alcuni elementi dell'insieme degli interi tra 1 e 10 la soddisfa; possiamo definire tale sottoinsieme attraverso i *predicati*:
pari(2), pari(4), pari(6),pari(8), pari(10).
essi significano "2 è pari", "4 è pari" etc...
ovvero che dell'*insieme universo* 1,2,...,10 i numeri 2,4,6,8,10 appartengono al sottoinsieme dei numeri pari
La *proposizione aperta* "X è pari" (pari(X) in PROLOG) è vera quando X vale 2,4,6,8,10 ; il suo insieme verità corrisponde cioè all'insieme dei numeri pari compresi tra 1 e 10.
Il programma consente di definire gli insiemi attraverso i predicati ed è in grado di ritrovare tutti gli elementi dell'*insieme verità*, cioè degli elementi che rendono vera la proposizione aperta.

?- assert(pari(2)).
Yes
?- assert(pari(4)).
Yes
?- assert(pari(6)).
Yes
?- assert(pari(8)).
Yes
?- assert(pari(10)).
Yes
?- pari(X).
X = 2 ;
X = 4 ;
X = 6 ;
X = 8 ;
X = 10 ;
No

Per ottenere una dopo gli elementi dell'insieme soluzione bisogna premere ripetutamente ";" fin quando l'algoritmo di esaustione dell'albero che rappresenta la base dati, non ha controllato o escluso tutte le interpretazioni.

Impostata in questo modo però la ricerca avviene sui soli elementi dell'inseme *pari* quello che viceversa interessa è definire l'insieme *primaDecina* e vedere se $\exists x$ appartenente a questo insieme e se $\forall x$ appartenente a questo insieme sia pari:

?- assert(primaDecina(1)).
Yes
?- assert(primaDecina(2)).
Yes
?- assert(primaDecina(3)).
Yes
?- assert(primaDecina(4)).
Yes
?- assert(primaDecina(5)).
Yes
?- assert(primaDecina(6)).
Yes
?- assert(primaDecina(7)).
Yes
?- assert(primaDecina(8)).
Yes
?- assert(primaDecina(9)).
Yes
?- assert(primaDecina(10)).
Yes

Ci chiediamo: **esiste** un x che appartenga alla prima decina e che sia pari?

?- primaDecina(X),pari(X).
X = 2 ;
X = 4 ;
X = 6 ;
X = 8 ;
X = 10 ;
No

La risposta è stata si e il programma ci ha anche fornito gli elementi che rendono vera la proposizione "X appartiene alla prima decina **e** è pari".

Ci chiediamo adesso se **qualsiasi** elemento della prima decina è pari:

?- pari(primaDecina(X)).
No

Il programma ha estratto l'insieme dei numeri appartenenti alla primaDecina ed è andato a controllare se erano tutti pari: giustamente la risposta è stata "falso" cioè **No**.