Esercitazione 4: PROLOG

Fondamenti di Intelligenza Artificiale

SAPIENZA
UNIVERSITÀ DI ROMA

A.A. 2022/2023

DIAG, Dipartimento di Ingegneria Informatica, Automatica, e Gestionale “Antonio Ruberti”.

Installazione swi-prolog (ubuntu/debian)

Tramite apt-get:

```
$ sudo apt-add-repository ppa:swi-prolog/stable
$ sudo apt-get install swi-prolog
```

Altri OS: vedere https://www.swi-prolog.org/Download.html

Repository codice esercitazioni

link: https://github.com/KRLGroup/FondamentiIA-2223

Download della repo con git (opzionale):

```
git clone https://github.com/KRLGroup/FondamentiIA-2223.git
```

Il codice (prolog) per questa esercitazione è in “`esercitazione04.pl`”

Alternativa: swi-prolog online

https://swish.swi-prolog.org/

Interfaccia che permette di scrivere programmi prolog ed eseguire query da browser.

Sessioni interattive (linux/macos)

Per iniziare una sessione interattiva: `swipl`

Viene presentato un prompt `?-` dal quale si possono caricare knowledge base, eseguire query, …

Maggiori info: https://www.swi-prolog.org/

Caricare knowledge base

Per caricare knowledge base da “`filename.pl`”:

```
?- [filename].
```

Query

Senza variabili (true/false):

```
?- member(1, [1,2,3]).
true.
```

Con variabili:

```
?- reverse([1,2,3], R).
R = [3,2,1]
```

Query con variabili

Se la query contiene variabili, la prima soluzione trovata viene stampata a schermo; da qui si può digitare:

- `;` o la barra spaziatrice, per trovare la prossima sostituzione
- `.` o il tasto invio, per terminare la query e tornare al prompt `?-`

Esercizio 1

Costruire gli alberi di ricerca per:

A) `member1(c,[a,c,b]).`
B) `plus1(Y,X,s(s(s(s(s(0))))))).`
C) `reverse1([a,b,c],X).`

Per le definizioni consultare le slide della lezione del 19/05.

Esercizio 2

Scrivere i seguenti programmi in PROLOG:

A) `pow1(B,E,Z)`, where `Z` is the result of `B` raised to the `E`
B) `minimum(X,Y,Z)`, using the predicate `lesseq1(N1,N2)` which is true when $X \leq Y$, false otherwise
C) `sum(N,Z)`, where `N` is a positive integer and `Z` is the result of summing up the first `N` numbers.

Esercizio 3

Scrivere i seguenti programmi in PROLOG:

A) `suffix(L1,L)`, true when `L1` is a suffix of `L`
B) `subset(A,B)`, true when `A` is a subset of `B`
C) `intersection(A,B,C)`, true when `C` is the intersection between `A` and `B`

Usando le liste per rappresentare gli insiemi.

Esercizio 4

Scrivere il programma `erdosnum(X,N)`, true se `N` è il numero di Erdos dell'autore `X`. Il *numero di Erdos* per un autore è così definito:

- il numero di Erdos di Paul Erdos è 0
- il numero di Erdos di un altro autore `X` è 1 sommato al minimo numero di Erdos tra tutti i suoi coautori.

Si usi il predicato `coauth(X,Y)` e la knowledge base in "`esercitazione04.pl`".