Εργαστήριο Λογικού Προγραμματισμού

Μανόλης Μαρακάκης, Καθηγητής
mmarak@cs.hmu.gr

Τμήμα Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών
Σχολή Μηχανικών
Ελληνικό Μεσογειακό Πανεπιστήμιο

Ενότητα 3: Μάθημα 4

3. Αναδρομή, Λίστες και Αριθμητική σε Prolog

- 3.1. Αναδρομή
- 3.2. Λίστες
  - ✓ 3.2.1. Δομή και Χρήση της Λίστα.
  - ✓ 3.2.2. Παραδείγματα Προγραμμάτων με Λίστες
- 3.3. Αριθμητική σε Prolog
- 3.4 Προγραμματιστικές Τεχνικές
- 3.5. Παράδειγμα: Οι 8 - Βασίλισσες

3. Λίστες και Αριθμητική σε Prolog

3.2. Λίστες: Παραδείγματα Προγραμμάτων με Λίστες

- ***Παράδειγμα 1:*** Το κατηγόρημα ***member(St, Lista)*** είναι αληθές εάν κάποιο στοιχείο **St** είναι μέλος στη λίστα **Lista**.

  π1: member(St, [St|Oura]).

  π2: member(St, [X|Oura]) :- member(St, Oura).

  Πρόγραμμα 3.3:*Μέλος λίστας*

  - Η πρόταση **π1** ελέγχει εάν το στοιχείο ***St*** είναι η **κεφαλή της λίστας**.
  - Η πρόταση **π2** ελέγχει εάν το στοιχείο ***St*** είναι **κάποιο από τα στοιχεία της ουράς της λίστας**.

- ***Παραδείγματα ερωτήσεων/στόχων:***

```
?- member( a, [ a, b, c ] ).
     yes
?- member( [ e ], [ a, b, [ c, d ], [ e ], [ f ] ] ).
     yes
?- member( X, [ a, b ] ).
     X= a;
     X=b;
     yes
```

3. Λίστες και Αριθμητική σε Prolog

3.2. Λίστες: Παραδείγματα Προγραμμάτων με Λίστες

- ***Παράδειγμα 2:*** Το κατηγόρημα **delete(X, Lista, Nlista)** χρησιμοποιείται για να σβήσουμε ένα στοιχείο **X** από την λίστα **Lista** δημιουργώντας μια νέα λίστα Nlista χωρίς το στοιχείο **X**.
- Για να προγραμματίσουμε τη σχέση **delete/3** διακρίνουμε τις εξής δύο περιπτώσεις:
  - **α.** Το στοιχείο **X** που θέλουμε να σβήσουμε είναι **η κεφαλή της λίστας Lista**,
  - **β.** Το στοιχείο **X** είναι κάποιο στοιχείο από τα στοιχεία στην **ουρά της λίστας Lista**.
- Στο **πρόγραμμα 3.4** η πρόταση **π1** αντιστοιχεί στην περίπτωση (**α**) ενώ η πρόταση **π2** στην περίπτωση (**β**).

```
π1: delete( X, [X|Oura], Oura ).
π2: delete( X, [Y|Oura], [Y|Oura1] ) :-  delete( X, Oura, Oura1).
```

Πρόγραμμα 3.4: ***Σβήσιμο της Πρώτης Επανάληψης Στοιχείου***

- **Παραδείγματα ερωτήσεων/στόχων**

```
?- delete( a, [1, a, b, c, 2, 3 ], L ).
        L = [1, b, c, 2, 3 ]
?- delete( a, [1, a, 2, a, 3, a ], L ).
        L = [1, 2, a, 3, a ];
        L = [1, a, 2, 3, a ];
        L = [1, a, 2, a, 3 ];
        no
```

3. Λίστες και Αριθμητική σε Prolog

3.2. Λίστες: Παραδείγματα Προγραμμάτων με Λίστες

- ***Παράδειγμα 3:*** Το κατηγόρημα **deleteAll(X, Lista, Nlista)** σβήνει όλες τις επαναλήψεις του στοιχείου **X** από την λίστα **Lista** δημιουργώντας την λίστα **Nlist**a. Διακρίνουμε τις εξής **τρεις περιπτώσεις**:
  - **α.** Η οριακή περίπτωση είναι όταν **η λίστα Lista είναι άδεια**.
  - **β.** Τα στοιχείο **X** είναι **η κεφαλή της λίστας Lista**.
  - **γ.** Το στοιχείο **X** είναι κάποιο στοιχείο της **ουράς της λίστας Lista**.
- Οι προτάσεις **π1**, **π2** και **π3** εξετάζουν τις περιπτώσεις **α**, **β**, και **γ** αντίστοιχα.

π1: deleteAll(_, [ ], [ ]).

π2: deleteAll(X,[X|Oura],Lista1) :- deleteAll(X,Oura,Lista1).

π3: deleteAll(X,[Y|Oura],[Y|Oura1]):-deleteAll(X,Oura,Oura1).

Πρόγραμμα 3.5:*Σβήσιμο όλων των Επαναλήψεων Στοιχείου*

- **Παραδείγματα ερωτήσεων/στόχων**
  - ?- deleteAll(a, [1, a, b, c, a, a], L).
  - L = [1, b, c]
  - ?- delete(a, [1, a, 2, a, 3, a], L).
  - L = [1, 2, 3]

3. Λίστες και Αριθμητική σε Prolog

3.2. Λίστες: Παραδείγματα Προγραμμάτων με Λίστες

- ***Παράδειγμα 4:*** Το κατηγόρημα ***append(Lista1, Lista2, Lista3)*** προεκτείνει την λίστα **Lista1** με την λίστα **Lista2**, σχηματίζοντας μια τρίτη λίστα **Lista3**.
  - Σαν **βασική περίπτωση** λαμβάνεται όταν **η Lista1 είναι κενή** (πρόταση **π1** του προγράμματος Πρόγραμμα 3.6).
  - Στη **γενική περίπτωση** αποσπά ένα-ένα τα στοιχεία της λίστας **Lista1** και τα τοποθετεί στην λίστα του τρίτου ορίσματος του **append/3** μέχρι η λίστα **Lista1** να μείνει κενή οπότε ισχύει η πρόταση **π1.**
  - Με την πρόταση **π1** όλα τα στοιχεία της λίστας **Lista2** τοποθετούνται στη λίστα του τρίτου ορίσματος του **append/3**.

π1: append([ ], Lista2, Lista2).

π2: append([X|Lista1], Lista2, [X|Lista3]) :- append(Lista1, Lista2, Lista3).

Πρόγραμμα 3.6: ***Προσάρτηση της Λίστας L2 στη Λίστα L1***

- ***Παραδείγματα ερωτήσεων/στόχων***

```
?- append( [ a, b, c ], [ k, m, n ], Lista ).
        Lista = [ a, b, c, k, l, n ]
?- append( [ a, [ b, c ] ], [ k, [ m ], [ n ] ], Lista ).
        Lista = [ a, [ b, c ] ,  k, [ m ], [ n ] ]
```

3. Λίστες και Αριθμητική σε Prolog

3.2. Λίστες: Παραδείγματα Προγραμμάτων με Λίστες

- ***Παράδειγμα 5:*** Το κατηγόρημα ***reverse(Lista, Alista)*** επιστρέφει την αντίστροφη λίστα **Alista**, της λίστας**, Lista**.
- Ως **βασική περίπτωση** λαμβάνεται όταν η λίστα που θέλουμε να αντιστρέψουμε είναι η κενή. Αυτό εκφράζει η πρόταση **π1**.
- Η λογική στην οποία στηρίζεται η αντιστροφή μιας λίστας όπως εκφράζεται με την πρόταση **π2** μπορεί να διατυπωθεί και ως εξής ΄Για να αντιστραφεί η λίστα **[St|Oura]** **πρώτα** αντίστρεψε την ουρά **Oura** της λίστας σχηματίζοντας την λίστα **L** και **κατόπιν** τοποθέτησε στο τέλος της αντεστραμμένης λίστας **L** την κεφαλή **St** της αρχικής λίστας **[St|Oura]**.

```
π1: reverse([ ], [ ] ).
π2: reverse([St|Oura], Alista ) :- reverse(Oura, L ),
                                   append(L, [St], Alista ).
```

Πρόγραμμα 3.7: *Αντιστροφή Λίστας*

- ***Παράδειγμα ερώτησης/στόχου***

```
?- reverse( [ a, b, c, d ], Alista ).
        Alista = [ d, c, b, a ]
?- reverse( [ [1], [2], [3],[4] ], Alista ).
        Alista = [ [4],[3],[2],[1]]
```

3. Λίστες και Αριθμητική σε Prolog

3.2. Λίστες: Παραδείγματα Προγραμμάτων με Λίστες

- ***Παράδειγμα 7:*** Το κατηγόρημα ***translate(Lista1, Lista2)*** μεταφράζει την λίστα των Ελληνικών λέξεων **Lista1** στη λίστα των Αγγλικών λέξεων **Lista2.**

```
translate( [ ], [ ]).
translate( [K1|O1], [K2|O2]) :-
        dictionary( K1, K2), translate(O1, O2).
dictionary(ena, one).
dictionary(duo, two).
dictionary(epi, times).
dictionary (ison, equal).
```

Πρόγραμμα 3.8:*Μετάφραση Ελληνικών Λέξεων σε Αγγλικές*

- ***<u>Παράδειγμα ερώτησης/στόχου</u>***

```
?- translate( [ena,epi,duo,ison,duo], Eng ).
        Eng = [one,times,two,equal,two ]
```

- ***Παράδειγμα 8:*** Το κατηγόρημα **selectionSort(L, SortedL)** είναι αληθές εάν η λίστα **SortedL** είναι η λίστα **L** με τα στοιχεία της ταξινομημένα κατ' αύξουσα σειρά.
- ***Αλγόριθμος:*** Η μέθοδος της **ταξινόμησης με επιλογή** (**selection sort**)
  - βρίσκει πρώτα το μικρότερο στοιχείο της **L** και το τοποθετεί στην ταξινομημένη λίστα **SortedL**.
  - Στη συνέχεια βρίσκει το μικρότερο από τα αναπομείναντα στοιχεία της **L** και το προσαρτεί στο τέλος της λίστας **SortedL**.
  - Αυτό το βήμα επαναλαμβάνεται μέχρι να αδειάσει η λίστα **L**.
- Το κατηγόρημα **selection(InputTail, H, MinEl, InputRest)** επιλέγει το μικρότερο **MinEl** στοιχείο από την λίστα **[H| InputTail]**. Επιστρέφει το μικρότερο στοιχείο **MinEl** καθώς και την λίστα **InputRest** με τα υπόλοιπα στοιχεία.

```
selectionSort([], []).
selectionSort([H|InputTail], Output) :-
        selection( InputTail, H, MinEl, InputRest ),    % επεξεργασία κεφαλής
        selectionSort(InputRest, OutputRest),
        append([MinEl], OutputRest, Output).
selection([], El, El, []).
selection([X|Xs], TestEl, El, [X|RestEl]) :-
                TestEl =< X,  selection(Xs, TestEl, El, RestEl).
selection([X|Xs], TestEl, El, [TestEl|RestEl]) :- selection(Xs, X, El, RestEl).
```

Πρόγραμμα 3.9:*Ταξινόμηση με Επιλογή*

- Το κατηγόρημα **selection(InputTail, H, MinEl, InputRest)** επιλέγει το μικρότερο **MinEl** στοιχείο από την λίστα **[H| InputTail]**. Επιστρέφει το μικρότερο στοιχείο **MinEl** καθώς και την λίστα **InputRest** με τα υπόλοιπα στοιχεία.
- ***Παράδειγμα ερώτησης/στόχου***

```
?- selectionSort([9,-5,4,15,-8,9,0,15], Lsorted).
        Lsorted = [-8,-5,0,4,9,9,15,15]
```

3. Λίστες και Αριθμητική σε Prolog

3.2. Λίστες: Παραδείγματα Προγραμμάτων με Λίστες

- ***Παράδειγμα 9:*** Το κατηγόρημα ***bubbleSort(L, Lsorted)*** είναι αληθές εάν η λίστα Lsorted είναι η λίστα L ταξινομημένη.
- ***Αλγόριθμος***: Η ταξινόμηση με ανταλλαγή **συγκρίνει τα γειτονικά στοιχεία τα οποία ανταλλάσσει εάν δεν ευρίσκονται στην σωστή σειρά**. Έτσι στο τέλος του πρώτου περάσματος το μεγαλύτερο στοιχείο βρίσκεται στην τελευταία θέση. Στο τέλος του δεύτερου περάσματος το μεγαλύτερο στοιχείο από τα εναπομείναντα (πλην του τελευταίου) βρίσκεται στην προτελευταία θέση και ούτω καθεξής. Οι ανταλλαγές επαναλαμβάνονται μέχρι να μην υπάρχει στοιχείο για ανταλλαγή.

```
bubbleSort(L, Lsorted) :- append(L1, [H1, H2|T2], L),
                          H2 < H1,
                          append(L1, [H2, H1|T2], NewL),
                          bubbleSort(NewL, Lsorted).

bubbleSort(L, L).
```

Πρόγραμμα 3.10: ***Ταξινόμηση με Ανταλλαγή Γειτονικών Στοιχείων***

- ***Παράδειγμα ερώτησης/στόχου***

```
?- bubbleSort([9,-5,4,15,-8,9,0,15], Lsorted).
        Lsorted = [-8,-5,0,4,9,9,15,15]
```

3. Λίστες και Αριθμητική σε Prolog

3.2. Λίστες: Παραδείγματα Προγραμμάτων με Λίστες

- ***Παράδειγμα 10:*** Το κατηγόρημα **writelist(Lista)** εκτυπώνει την λίστα **Lista** έτσι ώστε κάθε στοιχείο της λίστας **Lista** να γράφεται σε χωριστή γραμμή.

```
writelist( [] ).
writelist([X | L]) :-    write(X), nl,
writelist(L).
```

Πρόγραμμα 3.11:*Εκτύπωση Όλων των Στοιχείων Λίστας*

- ***Παράδειγμα ερώτησης/στόχου***

```
?- writelist([‘ΤΕΙ’, ‘Κρήτης’]).
    ΤΕΙ
    Κρήτης
```

Τέλος Διάλεξης

Ευχαριστώ!

Ερωτήσεις;