ΤΕΧΝΗΤΗ ΝΟΗΜΟΣΥΝΗ

6° Εξάμηνο

Λίστες και αναδρομικός Προγραμματισμός (Παραδείγματα)

Δημοσθένης Σταμάτης
http:/www.iee.ihu.gr/~demos
Τμήμα Μηχανικών Πληροφορικής & Ηλεκτρονικών Συστημάτων

Ορισμός Λίστας (υπενθύμιση)

Μία λίστα στην Prolog ορίζεται ως εξής:

`[ ]` για την περίπτωση που η λίστα είναι κενή

`[H|T]` για την περίπτωση που η λίστα έχει για κεφαλή το στοιχείο Η και ουρά της την Τ που είναι με τη σειρά της μία λίστα

Το σύμβολο **|** θεωρείται **ενθεματικό (infix)** συναρτησιακό σύμβολο και εφαρμόζεται αναδρομικά με την έννοια ότι η ουρά της λίστας **T** μπορεί με τη σειρά της να έχει τη μορφή:

T = [H2|T2] **(η T2 = [H3|T3] κ.ο.κ)**

Το μήκος μιας λίστας

Σκέφτομαι αναδρομικά !

- Γνωρίζω ότι το πρόβλημα είναι λυμένο για την ουρά **T** της λίστας **[H|T]**
- Άρα γνωρίζω το μήκος της ουράς: Length_of_T
- Συνεπώς προσθέτω 1 και βρίσκω το συνολικό μήκος Length.
- Οριακή περίπτωση η κενή λίστα με μήκος 0

```
length([ ], 0).
length([H|T], Length) :-
        length(T, Length_of_T),
        Length is Length_of_T + 1.
```


Το άθροισμα των ακέραιων στοιχείων μιας λίστας

Σκέφτομαι αναδρομικά !

- Γνωρίζω ότι το πρόβλημα είναι λυμένο για την ουρά **T** της λίστας **[H|T]**
- Άρα γνωρίζω το άθροισμα των στοιχείων της ουράς: Sum_of_T
- Συνεπώς προσθέτω το H και βρίσκω το συνολικό άθροισμα Sum.
- Οριακή περίπτωση η κενή λίστα με άθροισμα στοιχείων 0

```
sum([ ], 0).
sum([H|T], Sum) :-
        sum(T, Sum_of_T),
        Sum is Sum_of_T + H.
```

Ομοιότητες των αναδρομικών προγραμμάτων για λίστες

```
length([ ], 0).
length([H|T], Length) :-
        length(T, Length_of_T),
        Length is Length_of_T + 1.
```

```
sum([ ], 0).
sum([H|T], Sum) :-
        sum(T, Sum_of_T),
        Sum is Sum_of_T + H.
```

```
foo([ ], 0).
foo([H|T], S) :-
        foo(T, ST),
        S is ST + ???.
```


Το τελευταίο στοιχείο μιας λίστας

Σκέφτομαι αναδρομικά !

- Γνωρίζω ότι το πρόβλημα είναι λυμένο για την ουρά **T** της λίστας **[H|T]**
- Άρα γνωρίζω το τελευταίο στοιχείο της ουράς: Last
- Συνεπώς και το τελευταίο στοιχείο της αρχικής λίστας είναι Last
- Οριακή περίπτωση η λίστα με ένα στοιχείο που είναι και το τελευταίο της

```
last_of([H], H).
Last_of([H|T], Last) :-
        last_of(T, Last).
```

Το ελάχιστο στοιχείο μιας λίστας

Σκέφτομαι αναδρομικά !

- Γνωρίζω ότι το πρόβλημα είναι λυμένο για την ουρά **T** της λίστας **[H|T]**
- Άρα γνωρίζω το ελάχιστο στοιχείο στοιχείο της ουράς: Min_of_T
- Συνεπώς το ελάχιστο στοιχείο της αρχικής λίστας είναι είτε το H αν H=<Min_of_T, είτε το Min_of_T στην αντίθετη περίπτωση.
- Οριακή περίπτωση η λίστα με ένα στοιχείο που είναι και το ελάχιστό της.

```
min([H], H).
min([H|T], H) :-
        min(T, Min_of_T),
        H =<Min_of_T.          % 1η περίπτωση
min([H|T], Min_of_T) :-
        min(T, Min_of_T),
        H > Min_of_T.          % 2η περίπτωση
```


Το ελάχιστο στοιχείο μιας λίστας καλύτερα !

```
min([H], H).
min([H|T], H) :-
        min(T, Min_of_T),
        H =<Min_of_T.          % 1η περίπτωση
min([H|T], Min_of_T) :-
        min(T, Min_of_T),
        H > Min_of_T.          % 2η περίπτωση
```

```
minB([H], H).
minB([H|T], Min) :-
        min(T, Min_of_T),
        min_of_2(H, Min_of_T, Min).

min_of_2(X,Y,X) :- X =< Y.
min_of_2(X,Y,Y) :- X > Y.
```

Διαγραφή ενός στοιχείου από μια λίστα

Σκέφτομαι αναδρομικά !

- Γνωρίζω ότι το πρόβλημα είναι λυμένο για την ουρά **T** της λίστας **[H|T]**
- Άρα γνωρίζω πως διαγράφω το **X** από την ουρά **T** και ποια είναι η λίστα μετά τη διαγραφή: **NewT**
- Συνεπώς διατηρώντας το H ως κεφαλή έχω την τελική λίστα μετά τη διαγραφή: **[H|NewT]**
- Οριακή περίπτωση αυτό που θέλω να διαγράψω να βρίσκεται στην κεφαλή της λίστας

```
mydelete(X, [X|T], T).

mydelete(X, [H|T], [H|NewT]) :-
        mydelete(X, T, NewT).
```


Διαγραφή όλων των εμφανίσεων ενός στοιχείου από μια λίστα

Σκέφτομαι αναδρομικά !

- Στην περίπτωση αυτή συνεχίζω να διαγράφω το στοιχείο από όλες τις εμφανίσεις στην ουρά της αρχικής λίστας.
- Και εδώ φυσικά θεωρώ ότι το πρόβλημα είναι λυμένο για την ουρά της λίστας
- Οριακή περίπτωση όταν θέλω να διαγράψω ένα στοιχείο από μία κενή λίστα, αφήνω το αποτέλεσμα κενή λίστα

```
deleteAll(X, [], []).
deleteAll(X, [X|T], LT) :-
        deleteAll(X, T, LT).
deleteAll(X, [H|T], [H|LT]) :-
        X \= H,
        deleteAll (X, T, LT).
```

Ομοιότητες της mydelete με την mymember

Προσέξτε ότι τα 2 πρώτα ορίσματα της **mydelete** λειτουργούν ακριβώς με τον ίδιο τρόπο που λειτουργούν στην **mymember**.

Γενικός τρόπος αντιμετώπισης:

Βλέπω τι γίνεται όταν το στοιχείο που με ενδιαφέρει βρίσκεται στην κεφαλή της λίστας ή εναλλακτικά βρίσκεται στην ουρά της λίστας.

Θα μπορούσαμε να έχουμε:

```
mymember2(X,L) :- mydelete(X,L,_).
```

```
mydelete(X, [X|T], T).

mydelete(X, [H|T], [H|NewT]) :-
        mydelete(X, T, NewT).
```

```
mymember(X, [X|T]).

mymember(X, [H|T]) :-
        mymember(X, T).
```