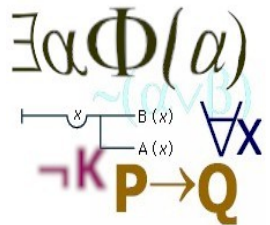


Λίστες

$$\exists \alpha \Phi(\alpha)$$


A collection of mathematical symbols including a universal quantifier $\forall x$, a lambda expression λx , a negation \neg , and a logical implication $P \rightarrow Q$.

Δομή

- Βασικές Έννοιες
 - Λίστες σαν Όροι, Αναρομικός Ορισμός Λίστας, Τρόπος Γραφής Λιστών στη Prolog
 - Ενοποίηση και Λίστες
- Αναδρομή και Λίστες
 - Παραδείγματα
- Ενσωματωμένα Κατηγορήματα
 - append/3, member/2
- Generate & Test

Δομές Δεδομένων

- Στην Prolog υπάρχει μόνο ΜΙΑ δομή: ο όρος (term):
 - functor(arg1, arg2, ...argn)
- Δεν υπάρχουν
 - sets, records, arrays, etc...
- Ο χειρισμός όρων με μεταβλητά ορίσματα είναι δύσκολος.

parent_of(nick, mary).

parent_of(nick, mary, john).

Λίστες

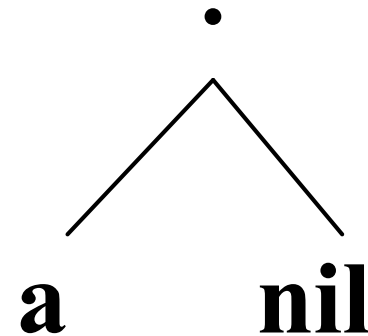
- Μοναδική δομή δεδομένων που προσφέρει η Prolog.
- Μια λίστα είναι ο όρος **./2**
 - Συναρτησιακό σύμβολο είναι η τελεία (.)
 - Arity 2
- Ορίσματα
 - Το πρώτο όρισμα είναι όρος.
 - Το δεύτερο όρισμα είναι λίστα.

Αναδρομικός Ορισμός

- Η κενή λίστα αναπαριστάται με *nil*.
- Μια **λίστα** είναι, είτε :
 - Η κενή λίστα *nil*.
 - Ένας όρος *./2* όπου το πρώτο όρισμα είναι οποιοσδήποτε όρος και το δεύτερο μια λίστα.

Παράδειγμα

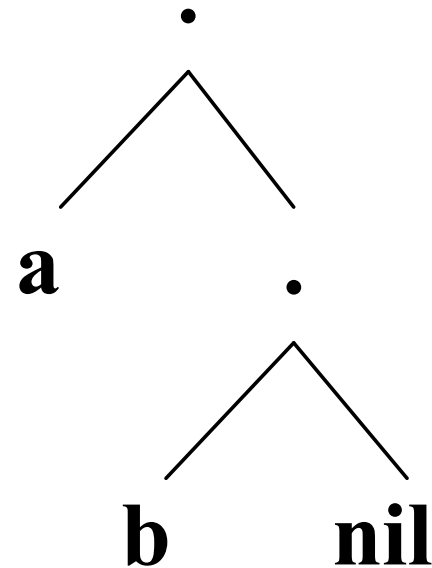
- Μια λίστα με ένα στοιχείο.
.(a, nil).



Παράδειγμα

Μια λίστα με δύο στοιχεία.

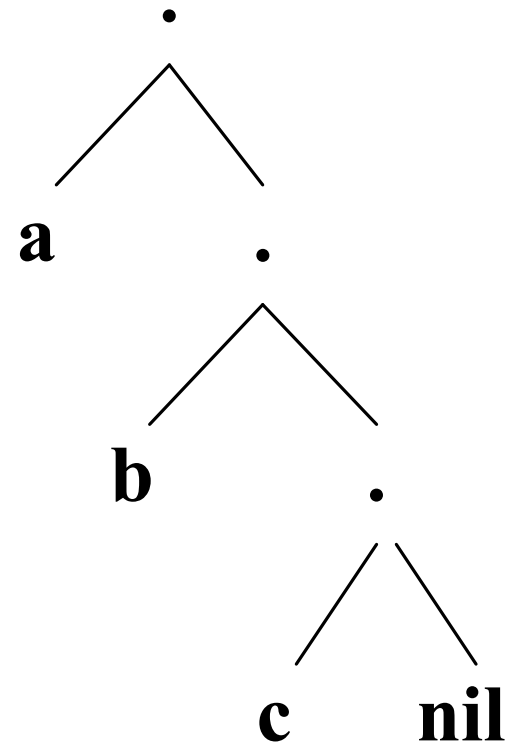
.(a, .(b, nil))



Παράδειγμα

Μια λίστα με τρία στοιχεία.

`.(a, .(b, .(c, nil)))`



Σημειογραφία (Notation)

- Σημειογραφία με παρενθέσεις δεν είναι εύχρηστη
- Η σημειογραφία της Prolog χρησιμοποιεί τις “[,]” και τον χαρακτήρα “|” .

nil \rightarrow [] .(*a*, *nil*) \rightarrow [*a*]

- 1 στοιχ.: [*car(daewoo)*]
- 2 στοιχ.: [*john, smith*]
- 3 στοιχ.: [*a, animal(elephant), animal(tiger)*]

Παραδειγμα: Ενδιαφέροντα Χρηστών

- Ενδιαφέροντα των χρηστών του κοινωνικού δικτύου.

likes(petros,[math,prolog,music,cinema]).

likes(ilias,[math,prolog,theatre,travel]).

likes(nikos,[prolog,travel,c,cooking]).

likes(demos,[music,cinema,theatre]).

likes(helen,[music,cinema,travel,cooking]).

likes(sofia,[physics,cinema,reading]).

Σημειογραφία (συν)

- Κεφαλή (Head)
 - Το πρώτο στοιχείο της λίστας
- Ουρά (Tail)
 - Τα στοιχεία εκτός του πρώτου.
- Head tail notation: [H|T]
 - Ο χαρακτήρας “|” διαχωρίζει την κεφαλή από την ουρά.

Παραδείγματα Λιστών

■ Σημειογραφία head tail

- Λίστα Ενός Στοιχείου

[a] *[a | []]*

- Λίστα Δύο Στοιχείων

[john, smith] *[john | [smith]]*

- Λίστα Τριών Στοιχείων

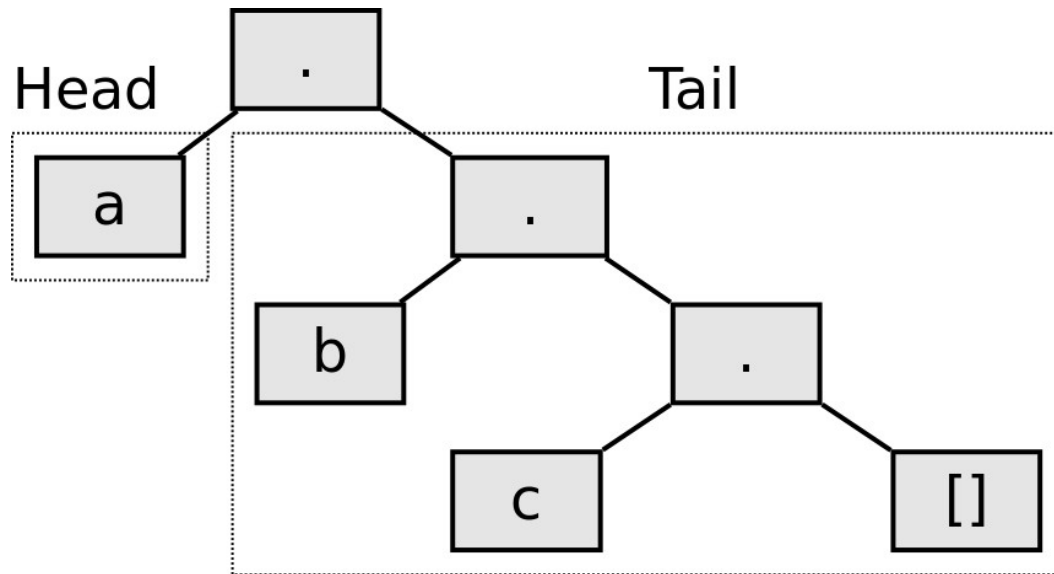
[a, animal(elephant), animal(tiger)]

[a | [animal(elephant), animal(tiger)]]

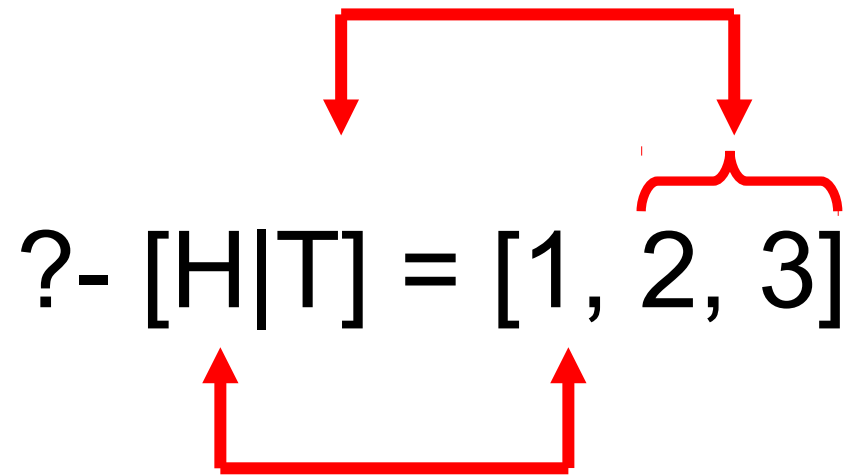
Παράδειγμα Κεφαλής-Ουράς

Μια λίστα με τρία στοιχεία.

$[a,b,c]$ *ή* $[a \mid [b,c]]$

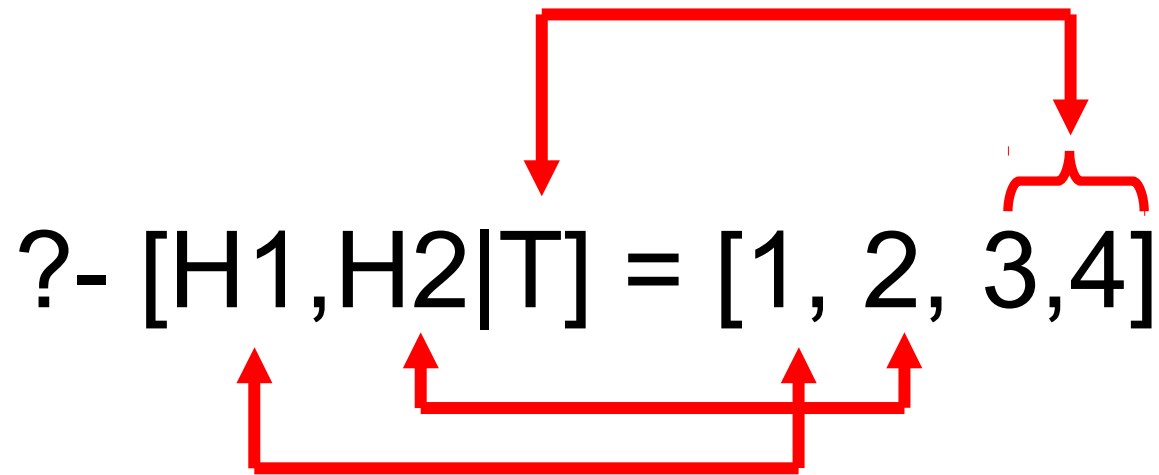


Παράδειγμα Ενοποίησης



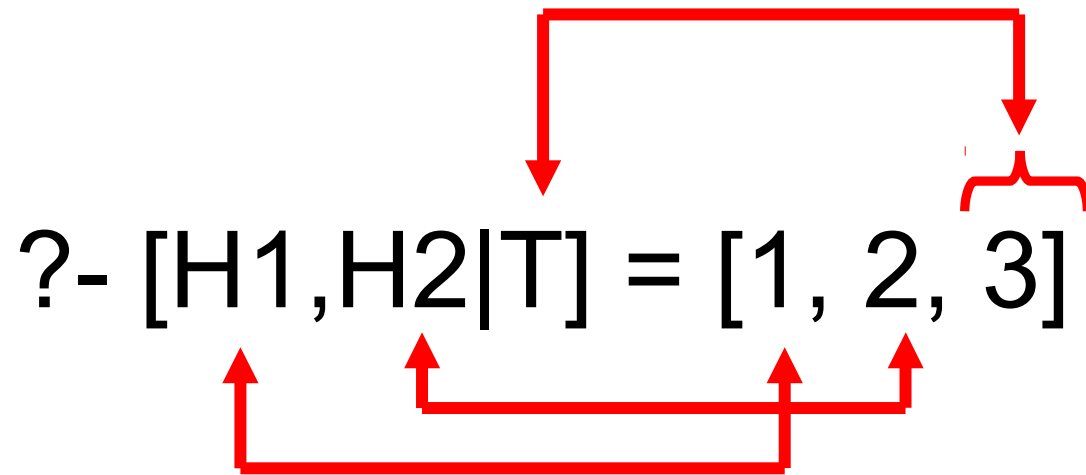
$H = 1, T = [2, 3]$

Παράδειγμα Ενοποίησης



$H1 = 1, H2=2, T = [3,4]$

Παράδειγμα Ενοποίησης



$H1 = 1, H2 = 2, T = [3]$

Αναδρομή σε λίστες

- Συνήθης περίπτωση επίλυσης προβλημάτων με λίστες.
- Επίλυση προβλήματος για λίστα [H|T]:
 - Θεωρώ ότι η λύση για κάποια υπολίστα υπάρχει.
 - Επιλύω το πρόβλημα για την λίστα [H|T] βάσει της προηγούμενης λύσης.
 - Δηλώνω βασικές περιπτώσεις.

Τελευταίο στοιχείο μιας λίστας

- Αναδρομικός Ορισμός

- Αν μια λίστα περιέχει μόνο ένα στοιχείο, τότε αυτό είναι και το τελευταίο.
- Εναλλακτικά, το τελευταίο στοιχείο μιας λίστας είναι το τελευταίο στοιχείο της ουράς της.

- Η περίπτωση (1):

last_element(Last,[Last]).

- Η περίπτωση (2):

*last_element(Last,[_Head|Tail]):-
last_element(Last, Tail).*

Το κατηγορημα `list_length/2`

- Να ορίσετε ένα κατηγορημα που βρίσκει το μήκος μιας λίστας.
- Αναδρομικός Ορισμός:
 - Η κενή λίστα έχει μήκος 0.
 - Μια λίστα έχει μήκος ίσο με το μήκος της ουράς της + 1.

Prolog Code

list_length([],0).

Η κενή λίστα
έχει μήκος 0.

*list_length([H|T] ,Len):-
list_length(T, TLen),
Len is TLen + 1.*

Μια λίστα έχει
μήκος ίσο με το
μήκος της
ουράς της + 1

Μέγιστος Ακέραιος σε μια Λίστα

- Αρχική Προσέγγιση (κεφαλή-ουρά)
 - Σε μια λίστα ενός στοιχείου, μέγιστο είναι αυτό το στοιχείο.
 - Εάν το μέγιστο στοιχείο της ουράς είναι **μεγαλύτερο** από την κεφαλή τότε αυτό είναι το μέγιστο στοιχείο.
 - Αλλιώς μέγιστο στοιχείο είναι η κεφαλή.

max/2 Predicate

- Prolog Code for the Maximum Num of a List.

max([X],X).

*max([H | T] ,H):-
 max(T, MT),
 H > MT.*

*max([H | T], MT):-
 max(T, MT),
 H =< MT.*

Εναλλακτική μέθοδος

- Προηγούμενος ορισμός δεν είναι αποδοτικός.
- Σύγκριση τρέχοντος στοιχείου με το μεγαλύτερο που βρέθηκε μέχρι τώρα.
- Όταν η λίστα εξαντληθεί επέστρεψε το μέγιστο στοιχείο.

max/2 Predicate: Alternative

```
max([ H | T ],Max):-  
    max_aux( T, H, Max).
```

```
max_aux([],MaxSoFar,MaxSoFar).
```

```
max_aux( [H | T] ,MaxSoFar, Max):-  
    H > MaxSoFar,  
    max_aux( T, H, Max).
```

```
max_aux( [ H | T] ,MaxSoFar, Max):-  
    H =< MaxSoFar,  
    max_aux( T, MaxSoFar, Max).
```


Ανάστροφη λίστα

- Ορίστε ένα κατηγορημα που πετυχαίνει όταν το πρώτο του όρισμα είναι η ανάστροφη λίστα του δευτέρου του ορίσματος.

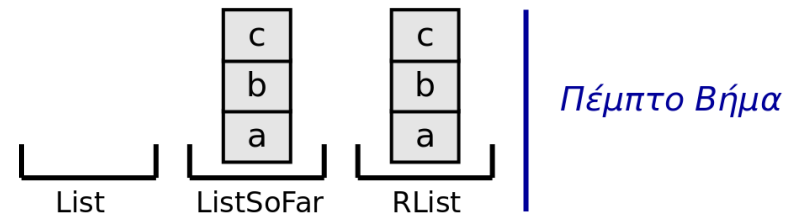
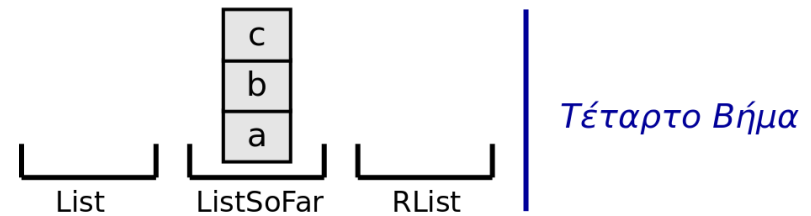
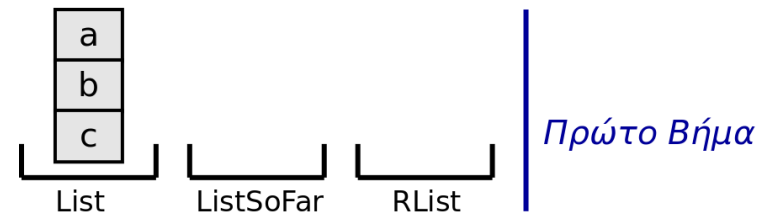
?- reverse([a, b, c], [c, b, a]).

yes

?- reverse([1, 2, 3], List).

List = [3, 2, 1].

Βασική Ιδέα



Κατηγορία reverse_list/2

■ Prolog Code

```
reverse_list( List, RList):-  
    reverse_list( List, [], RList).
```

```
reverse_list([], List, List).
```

```
reverse_list([H | T ], ListSoFar, RList):-  
    reverse_list(T,[H | ListSofar], RList ).
```

Μήκος Even/Odd

- *Evenlength/1*

- ☐ *Evenlength(List)* πετυχαίνει αν η λίστα *List* έχει άρτιο (*even*) αριθμό ορισμάτων (*0* θεωρείται *even*).

- *Oddlength/1*

- ☐ *Oddlength(List)* πετυχαίνει όταν η λίστα *List* έχει περιττό αριθμό ορισμάτων.

- Ορίστε δύο κατηγορήματα που να υλοποιούν το παραπάνω.

Note: **ΔΕΝ** μπορείτε να χρησιμοποιήσετε αριθμητικές πράξεις.

Ενσωματωμένα Κατηγορήματα

Ενσωματωμένα Κατηγορήματα

- Κλασικά Ενσωματωμένα κατηγορήματα:
 - Μήκος Λίστας (*αναφέρθηκε*)
 - Μέλος μιας Λίστας
 - Διαγραφή Στοιχείου
 - Συνένωση Λιστών

Μέλος Λίστας

■ Παραδείγματα:

?- member(1,[1,4,3])

yes

?- member(3,[1,2,3,4,5])

yes

?- member(2,[1,3,4])

no

Μέλος μιας Λίστας

- Κατηγορήμα `member/2`
 - *`member_list(Element, List)`*
- Αναδρομικός Ορισμός
 - Το στοιχείο είναι η κεφαλή της λίστας.
 - Το στοιχείο είναι μέλος της ουράς της Λίστας.

Prolog Code

- Κατηγορήμα ***member_list/2***

member_list(X, [X | Tail]).

*member_list(X, [Head | Tail]):-
member_list(X, Tail).*

- ECLiPSe έχει το κατηγορήμα ***member/2***.
 - ΔΕΝ απαιτείται να το ορίσετε ξανά.

Χρήση

- Όπως τα περισσότερα κατηγορήματα, μπορεί να χρησιμοποιηθεί με τουλάχιστον 2 τρόπους.

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

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

- Στη λογική δεν υπάρχει διαχωρισμός ορισμάτων input - output.
 - ...στις περισσότερες περιπτώσεις.

Παραδειγμα: Ενδιαφέροντα Χρηστών (revisited)

- Ενδιαφέροντα των χρηστών του κοινωνικού δικτύου.

likes(petros,[math,prolog,music,cinema]).

likes(ilias,[math,prolog,theatre,travel]).

likes(nikos,[prolog,travel,c,cooking]).

likes(demos,[music,cinema,theatre]).

likes(helen,[music,cinema,travel,cooking]).

likes(sofia,[physics,cinema,reading]).

Hands on!

- Να ορίσετε το κατηγορημα $\text{interest}(\text{User}, \text{Interest})$ το οποίο πετυχαίνει όταν ο User έχει στα ενδιαφέροντά του το Interest. Κατά την οπισθοδρόμηση θα επιστρέφει όλα τα ενδιαφέροντα του χρήστη. Για παράδειγμα:
 - $?- \text{interest}(\text{petros}, X).$
 - $X = \text{math}$
 - Yes ;
 - $X = \text{prolog}$
 - Yes ;
 - ...

Διαγραφή ενός στοιχείου

- Πόσα ορίσματα?
- Κατηγορήμα **delete_one/3**
 - *delete_one(Element, List, DelList)*

- Παραδείγματα:

?- *delete_one(a, [a, b, c], L).*

L = [b, c]

Yes

?- *delete_one(b, [a, b, c], L).*

L = [a, c]

yes

Διαγραφή ενός στοιχείου

- Κατηγορήμα **delete_one/3**
 - ***delete_one(Element, List, DelList),***
- Αναδρομικός Ορισμός
 - Όταν το στοιχείο προς διαγραφή είναι στην κεφαλή της λίστας, τότε το αποτέλεσμα είναι η ουρά.
 - Έστω ότι έχει διαγραφεί το στοιχείο από την ουρά της λίστας, οπότε το αποτέλεσμα είναι η κεφαλή + νέα ουρά .

Prolog Code

- Κατηγορήμα **delete_one/3**

delete_one(X, [X | Tail], Tail).

*delete_one(X, [H | Tail],[H | NewList):-
delete_one(X, Tail, NewList).*

- Η EclIPSe έχει το κατηγορήμα **delete/3**.
ΔΕΝ απαιτείται να το ορίσετε ξανά.

Συνένωση Λιστών

- Πόσα ορίσματα;
- Κατηγορήμα **concatenate/3**

?- concatenate([1, 2, 3], [a, b, c], L).

L = [1, 2, 3, a, b, c]

Yes

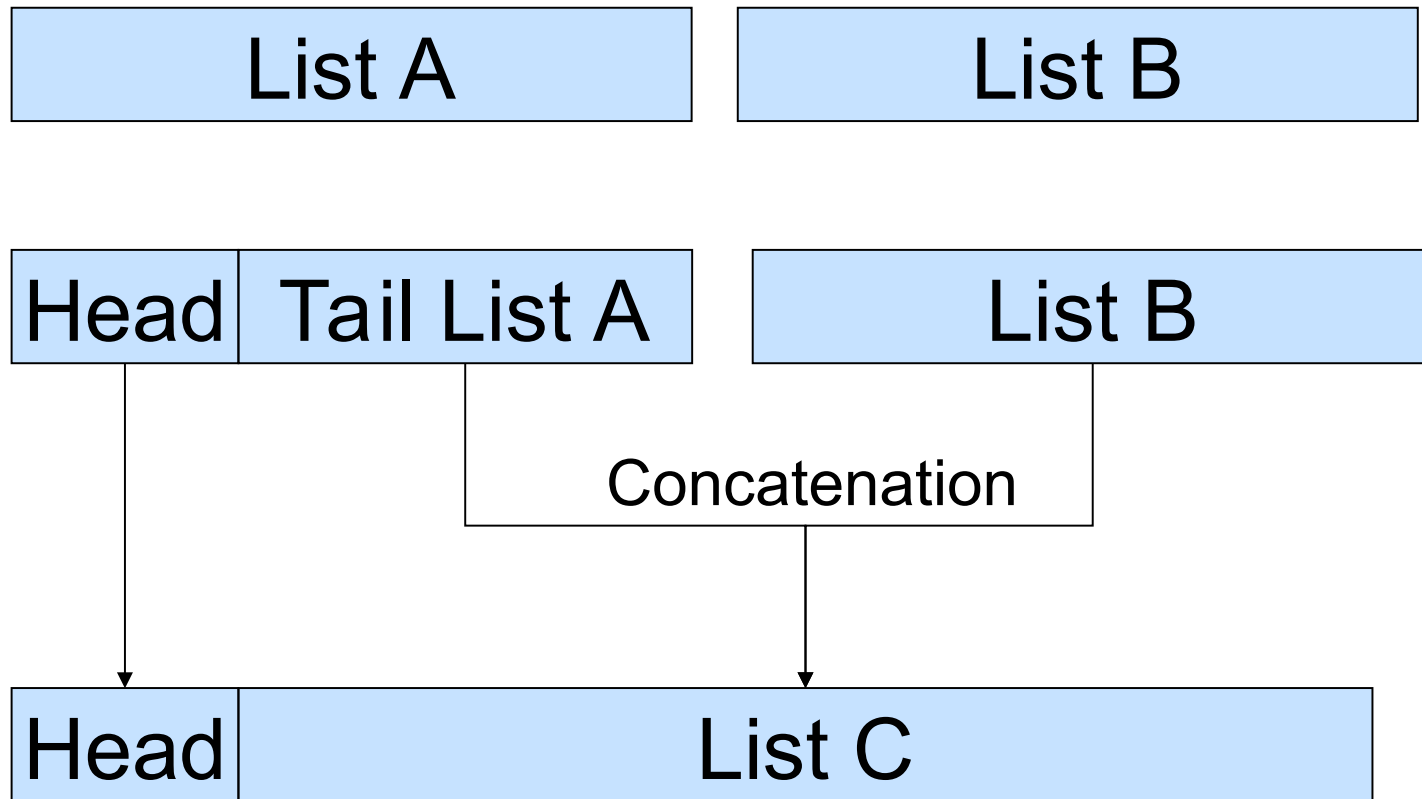
?- concatenate([1], [a, b, c], L).

L = [1, a, b, c]

Συνένωση Λιστών

- Κατηγορήμα **concatenate/3 (append/3)**
- Αναδρομικός ορισμός
 - Η συνένωση μιας λίστας με την κενή είναι η ίδια η λίστα.
 - Η συνένωση μιας λίστας A με μια λίστα B, “ισούται” με την τοποθέτηση της κεφαλής της A στην λίστα που προκύπτει από την συνένωση της ουράς της A με την B.

Συνένωση Λιστών



Prolog Code

- Κατηγορημα ***concatenate/3***

concatenate([], List, List).

*concatenate([H | Tail], List, [H | CList]):-
concatenate(Tail, List, CList).*

- AKA *append/3*

- Η υλοποίηση της *append/3* υπάρχει στις περισσότερες υλοποιήσεις της Prolog.

list_append([a,b],[c,d],L)

$\{ X/a, Tail/[b], L2 / [b,c],$
 $L / [a|[b,c,d]] \}$

list_append([b],[c,d],L3)

$\{ X'/b, Tail'/[], L2' / [c,d],$
 $L3 / [b|[c,d]] \}$

list_append([], [c,d], L3')

$\{ L3' / [c,d] \}$

Success

$L = [a,b,c,d]$

Κώδικας Prolog

list_append([],L2,L2).

list_append([X|Tail],L2,[X|L3]):-
list_append(Tail,L2,L3).

Χρήση της Append

- Μπορεί να χρησιμοποιηθεί με πολλούς τρόπους!
 - Διαχωρισμός Λιστών
 - Όπως και τα περισσότερα Prolog κατηγορήματα!!!

Παραγωγή και Δοκιμή (Generate & Test)

Generate and Test Στρατηγική

- Η απλούστερη λογική για την επίλυση προβλημάτων.
 - Παραγωγή μιας λύσης (generate)
 - Έλεγχος ορθότητας
 - Εάν δεν είναι ορθή η παραγόμενη λύση τότε παραγωγή εναλλακτικής.
- “Απλοϊκή” προσέγγιση στην επίλυση προβλημάτων.
- Πολύ εύκολη υλοποίηση σε Prolog
 - Εκμετάλλευση backtracking!

Επίλυση Αριθμητικών Εξισώσεων

- Έστω ότι X και Y ανήκουν στο $[1..10]$
- Βρείτε τα X, Y που ικανοποιούν τις παρακάτω ανισο-ισότητες:
 - $X - Y = 5$
 - $X + Y < 15$
- Χρήση `member/2` για ανάθεση τιμών στις μεταβλητές.

Prolog Code

■ Κατηγορία *pairs/2*

pairs(X, Y):-

member(X, [1,2,3,4,5,6,7,8,9,10]),

member(Y, [1,2,3,4,5,6,7,8,9,10]),

5 is X - Y ,

T is X + Y,

T < 15.

Δύο “σχεδόν” ίδιες Λίστες

- Γράψτε ένα κατηγορημα το οποίο πετυχαίνει όταν οι δύο λίστες στα ορίσματά του διαφέρουν μόνο κατά ένα στοιχείο.

*almost_identical(L1, L2):-
 delete_one(_,L1,L2).*

Υπολίστα μιας Λίστας

- Υλοποιείστε ένα κατηγορήμα που πετυχαίνει αν η λίστα A είναι υπολίστα της B.

?- sublist([2,3],[1,2,3,4,5])

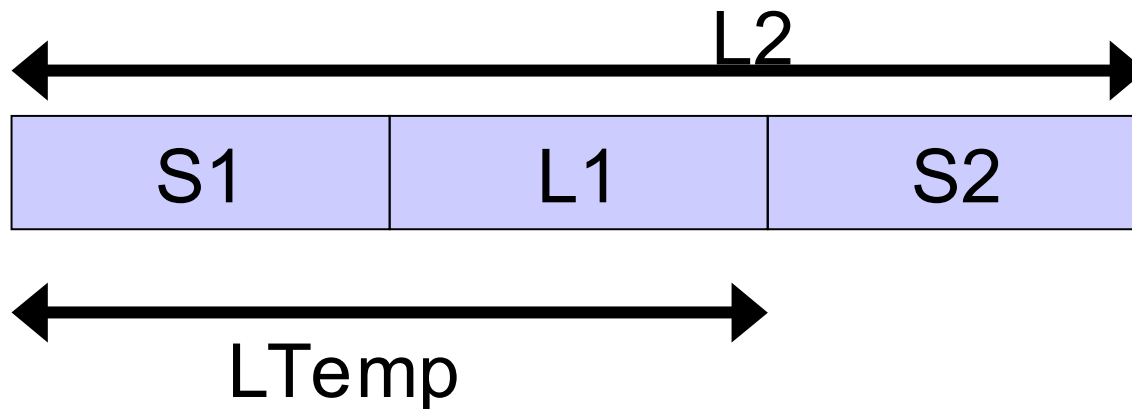
yes

Sublist/2

sublist(L1,L2):-

append(Ltemp, _S2, L2),

append(_S1, L1,Ltemp).



Δομή

- Βασικές Έννοιες
 - Λίστες σαν Όροι, Αναφορικός Ορισμός Λίστας, Τρόπος Γραφής Λιστών στη Prolog
 - Ενοποίηση και Λίστες
- Αναδρομή και Λίστες
 - Παραδείγματα
- Ενσωματωμένα Κατηγορήματα
 - append/3, member/2
- Generate & Test