Εργαστήριο Λογικού Προγραμματισμού

Μανόλης Μαρακάκης, Καθηγητής
mmarak@cs.hmu.gr

Τμήμα Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών
Σχολή Μηχανικών
Ελληνικό Μεσογειακό Πανεπιστήμιο

Κατασκευή Μεγάλων Προγραμμάτων σε Prolog

Ενότητα 8.5:

8. Προγραμματιστικές Τεχνικές
8.5 Δομές Δεδομένων σε Prolog

8. Προγρ/κές Τεχνικές: Δομές Δεδομένων σε Prolog.

- 8.5 Δομές Δεδομένων σε Prolog
  - 8.5.1 Εισαγωγή
  - 8.5.2 Ακολουθίες/Sequences
  - 8.5.3 Σύνολα/Sets
  - 8.5.4 Στοίβες/Stacks
  - 8.5.5 Ουρές/Queues

8.5. Δομές Δεδομένων σε Prolog

8.5.3. Σύνολα/Sets: Εισαγωγή

- Θα παρουσιάσουμε τα σύνολα ως ένα μια δομή δεδομένων καθώς και την υλοποίηση των πράξεων αυτής της δομής. Θεωρούμε ότι ο αναγνώστης γνωρίζει τη θεωρία των συνόλων και ειδικά τις πράξεις των συνόλων.
- Ένα ***σύνολο*** είναι μια συλλογή από καλά-ορισμένα, διακριτά αντικείμενα τα οποία μπορούμε να χειριστούμε ως μια οντότητα. Κάθε αντικείμενο της συλλογής λέγεται ***στοιχείο*** ή **μέλος** του συνόλου.
- Για ένα αντικείμενο **x** και ένα σύνολο **S**,
  - εάν το **x** είναι στοιχείο του συνόλου **S**, αυτό γράφεται ως **x** $\in$ **S**,
  - εάν το **x** δεν είναι μέλος του συνόλου **S** αυτό γράφεται ως **x** $\notin$ **S**.
- Υπάρχει ένα ειδικό σύνολο, το άδειο σύνολο, το οποίο συμβολίζεται {} ή ∅.

8.5. Δομές Δεδομένων σε Prolog

8.5.3. Σύνολα/Sets: Εισαγωγή

- Τα ιδιαίτερα χαρακτηριστικά των συνόλων είναι τα εξής:
  - 1) Δεν υπάρχουν επαναλήψεις στοιχείων, δηλαδή τα σύνολα **S1={a,b}** και **S2= {a,b,b,b}** είναι ίδια.
  - 2) Τα στοιχεία ενός συνόλου δεν έχουν σειρά, δηλαδή τα σύνολα **S1={a,b}** και **S2= {b,a}** είναι ίδια.
- Θα αναφέρουμε συνοπτικά και θα ορίσουμε περιγραφικά τις βασικές **ιδιότητες, σχέσεις** και **πράξεις** των συνόλων.

8.5. Δομές Δεδομένων σε Prolog

8.5.3. Σύνολα/Sets: Εισαγωγή

❑ Ιδιότητες και σχέσεις των συνόλων

- ***Ίσα σύνολα***. Δύο σύνολα **A** και **B** λέγονται **ίσα**, συμβολίζεται ως **A = B**, **εάν και μόνο εάν** κάθε στοιχείο του **A** είναι και στοιχείο του **B** και αντίστροφα, **διαφορετικά** λέγονται **άνισα** και συμβολίζεται ως **A ≠ B**.
- ***Σχέση υποσυνόλου***: Έστω δύο σύνολα **A** και **B**. Το σύνολο **A** είναι **υποσύνολο** του **B** και συμβολίζεται ως **A⊆B** **εάν και μόνο εάν** κάθε στοιχείο του **A** είναι και στοιχείο του **B**. Αυτή η σχέση μπορεί να γραφτεί και ως **B⊇A** που σημαίνει ότι το **B** είναι **υπερσύνολο** του **A**.
  - **Εάν** το σύνολο **A** είναι υποσύνολο του **B** αλλά **A ≠ B**, δηλαδή αν υπάρχει τουλάχιστον ένα στοιχείο του **B** το οποίο να μην ανήκει στο **A**, τότε λέμε ότι το σύνολο **A** είναι **γνήσιο υποσύνολο** του **B** και το συμβολίζουμε ως εξής **A ⊂ B**.
- **Δυναμοσύνολο**. Το **δυναμοσύνολο** του συνόλου **A** συμβολίζεται ως **℘(A)** και είναι το σύνολο με στοιχεία όλα τα υποσύνολα του συνόλου **A**.
- **Πληθικότητα**. **Πληθικότητα** ή **πληθάριθμος** ή **πληθικός αριθμός** ενός συνόλου **A** είναι το πλήθος των στοιχείων του.

8.5. Δομές Δεδομένων σε Prolog

8.5.3. Σύνολα/Sets: Εισαγωγή

- **Πράξεις συνόλων**
  - ***Ένωση συνόλων.*** Έστω τα σύνολα **A** και **B**, η ένωση των **A** και **B** είναι το σύνολο **A∪B** το οποίο ορίζεται ως εξής: **A∪B** = **{x| x ∈ A ∨ x ∈ B}.**
  - ***Τομή συνόλων***. Έστω τα σύνολα **A** και **B**, η τομή των **A** και **B** είναι το σύνολο **A∩B** το οποίο ορίζεται ως εξής: **A∩B** = **{x| x ∈ A ∧ x ∈ B}.**
  - ***Διαφορά συνόλων***. Έστω τα σύνολα **A** και **B**, η διαφορά των **A** και **B** είναι το σύνολο **A-B** το οποίο ορίζεται ως εξής: **A-B** = **{x| x ∈ A ∧ x ∉ B}.**
- Τα σύνολα θα υλοποιηθούν με λίστες της Prolog.
- Στη συνέχεια παρουσιάζουμε για κάθε **ιδιότητα** ενός συνόλου, και για κάθε **σχέση** και **πράξη** μεταξύ συνόλων τον περιγραφικό ορισμό της καθώς και την υλοποίηση της σε Prolog.

8.5. Δομές Δεδομένων σε Prolog

8.5.3. Σύνολα/Sets: Ορισμός και υλοποίηση πράξεων .

- Το κατηγόρημα **empty_set(S:set(T))** είναι αληθές εάν **S** είναι το άδειο σύνολο, mode(a).

```
empty_set([]).
```

Πρόγραμμα 8.39: Άδειο σύνολο.

- Το κατηγόρημα **member_set(X:T, S:set(T))** είναι αληθές εάν το στοιχείο **X** είναι μέλος του συνόλου **S**, mode(a,g)

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

Πρόγραμμα 8.40: Μέλος συνόλου.

8.5. Δομές Δεδομένων σε Prolog

8.5.3. Σύνολα/Sets: Ορισμός και υλοποίηση πράξεων .

- Το κατηγόρημα **set_cardinality(S:set(T1), N:natural)** είναι αληθές εάν **N** είναι η πληθικότητα του συνόλου **S**, mode(g,a).

```
set_cardinality([], 0).
set_cardinality([H|T], N) :-  \+ member_set(H, T),
                              set_cardinality(T, N1),  N is N1+1.
set_cardinality([H|T], N) :-  member_set(H, T), set_cardinality(T, N).
```

Πρόγραμμα 8.41: Πληθικότητα συνόλου.

- Το κατηγόρημα **insert_elem_set(X:T, S1:set(T), S2:set(T))** είναι αληθές εάν **S2** είναι το σύνολο **S**1 συν το στοιχείο **X**, mode(g,g,a).Επειδή η σειρά των στοιχείων στα σύνολα δεν είναι σημαντική ένα στοιχείο μπορεί να καταχωρηθεί οπουδήποτε σε μια λίστα είτε στην αρχή, ή στο μέσο, ή στο τέλος. Επιλέχθηκε η **αρχή** για πιο αποτελεσματική υλοποίηση.

```
insert_elem_set(X, S1, S1) :-    member_set(X, S1).
insert_elem_set(X, S1, [X|S1]) :-      \+ member_set(X, S1).
```

Πρόγραμμα 8.42: Καταχώρηση στοιχείου σε σύνολο.

8.5. Δομές Δεδομένων σε Prolog

8.5.3. Σύνολα/Sets: Ορισμός και υλοποίηση πράξεων .

- Το κατηγόρημα **delete_elem_set(X:T,S1:set(T),S2:set(T))** είναι αληθές εάν το σύνολο **S2** έχει τα στοιχεία του συνόλου **S1** εκτός από το στοιχείο **X**, mode(g,g,a).

```
delete_elem_set(X, [X|T1], T1).

delete_elem_set(X, [H1|T1], [H1|T2]) :- delete_elem_set(X, T1, T2).
```

Πρόγραμμα 8.43: Διαγραφή στοιχείου από σύνολο.

- Το κατηγόρημα **difference(S1:set(T),S2:set(T),S3:set(T))** είναι αληθές εάν **S3** είναι το σύνολο της διαφοράς των συνόλων **S1** και **S2** mode(g,g,a).

```
difference(S1, S2, S3) :-   difference([], S2, []).

difference([H1|T1], S2, S3) :-  member_set(H1,S2),
                                difference(T1, S2, S3).

difference([H1|T1],S2, [H1|T3]) :-  \+ member_set(H1,S2),
                                difference(T1,S2, T3).
```

Πρόγραμμα 8.44: Διαφορά συνόλων.

8.5. Δομές Δεδομένων σε Prolog

8.5.3. Σύνολα/Sets: Ορισμός και υλοποίηση πράξεων .

- Το κατηγόρημα **union(S1:set(T), S2:set(T), S3:set(T))** είναι αληθές εάν **S3** είναι η ένωση των συνόλων **S1** και **S2**, mode(g,g,a).

```
union([], S2, S2).
union([H1|T1], S2, [H1|T3]) :- \+ member_set(H1, S2),  union(T1, S2, T3).
union([H1|T1], S2, S3) :-  member_set(H1, S2),   union(T1, S2, S3).
```

Πρόγραμμα 8.45: Ένωση συνόλων.

- Το κατηγόρημα **intersection(S1:set(T), S2:set(T), S3:set(T))** είναι αληθές εάν **S3** είναι η τομή των συνόλων **S1** και **S2**, mode(g,g,a).

```
intersection([], S2, []).
intersection([H1|T1], S2, [H1|T3]) :- member_set(H1,S2),
                                      intersection(T1, S2, T3).
intersection([H1|T1], S2, S3) :-  \+ member_set(H1, S2),
                                      intersection(T1, S2, S3).
```

Πρόγραμμα 8.46: Τομή συνόλων.

8.5. Δομές Δεδομένων σε Prolog

8.5.3. Σύνολα/Sets: Ορισμός και υλοποίηση πράξεων .

- **Παράδειγμα 8.11.** Θα μελετήσουμε την ταξινόμηση με παρεμβολή πως μπορούμε να την υλοποιήσουμε χρησιμοποιώντας **σύνολα** και **ακολουθίες**.
- Το κατηγόρημα «**insert_sort(S1:set(T), S2:seq(T))**» είναι αληθές εάν η ακολουθία **S2** έχει τα στοιχεία του συνόλου **S1** ταξινομημένα, **mode(g,a)**.
  - Ο αλγόριθμος της ταξινόμησης με παρεμβολή παίρνει ένα-ένα τα στοιχεία από το σύνολο **S1** και τα τοποθετεί στην ακολουθία **S2**.
  - Το επιπλέον βοηθητικό κατηγόρημα «**ins_elem/3**» που χρησιμοποιείται για την υλοποίηση του **insert_sort/2** ορίζεται ως εξής. Το κατηγόρημα «**ins_elem(E:T,Q1:seq(T), Q:seq(T))**» είναι αληθές εάν η ακολουθία **Q** είναι η ταξινομημένη ακολουθία **Q1** αυξημένη με το στοιχείο **E** χωρίς να επηρεαστεί η υπάρχουσα σειρά, **mode(g,g,a)**.

8.5. Δομές Δεδομένων σε Prolog

8.5.3. Σύνολα/Sets: Ορισμός και υλοποίηση πράξεων .

- **insert_sort(S1, S2)** :- empty_set(S1), empty_seq(S2). % Πράξεις α) συνόλων β) ακολουθιών

  insert_sort(S1, S2) :- \+ empty_set(S1), % Πράξη συνόλων

  member_set(Elem,S1), delete_elem_set(Elem, S1, RestS1), % Πράξεις συνόλων

  insert_sort(RestS1, RestS2), **ins_elem(Elem,RestS2,S2).** % Νέο κατηγόρημα

- **ins_elem(E,Q1, Q)** :- empty_seq(Q1), seq_cons(Q1,E, Q). % Πράξεις ακολουθιών

  ins_elem(E,Q1, Q) :- head(Q1, V1), % Πράξη ακολουθιών

  E @=< V1, seq_cons(Q1, E, Q). % Πράξεις α) σύγκρισης όρων, β) ακολουθιών

  ins_elem(E,Q1, Q) :- head(Q1,V), E @> V, tail(Q1,Q1a),% Πράξεις ακολουθιών & συγκρ. όρων

  ins_elem(E,Q1a, Qa), seq_cons(Qa,V, Q). % Πράξη ακολουθιών

Πρόγραμμα 8.47: Υλοποίηση με σύνολα της ταξινόμησης με παρεμβολή.

- **Στόχοι:** Το πρόγραμμα ακολουθεί τη στάνταρντ σειρά ταξινόμηση όρων της Prolog.
- ?- insert_sort([1,5,-8,5,-3],S2). S2 = [-8,-3,1,5,5] ? yes
- ?- insert_sort([1,5,-3],S2). S2 = [-3,1,5] ? yes
- ?- insert_sort([1,5,-3,-8,99,2],S2). S2 = [-8,-3,1,2,5,99] ? yes
- ?- insert_sort([1,5,-3,-8,a,c,ab,2],S2). S2 = [-8,-3,1,2,5,a,ab,c] ? yes
- ?- insert_sort([X,Y,3.2,5,-3,a,c,ab,2],S2). S2 = [X,Y,3.2,-3,2,5,a,ab,c] ? yes

8.5. Δομές Δεδομένων σε Prolog

8.5.4. Στοίβες: Εισαγωγή

- Μια πολύ γνωστή και απλή δομή αλλά με πολλές χρήσεις σε εφαρμογές της Τεχνητής Νοημοσύνης είναι η **στοίβα *(stack)*.**
- Οι ***στοίβες*** **είναι διατεταγμένες λίστες** στις οποίες οι καταχωρήσεις και οι αφαιρέσεις των στοιχείων γίνονται από την μία άκρη η οποία ονομάζεται **κορυφή**.
- Σε μια στοίβα $\Sigma = <\sigma_1, \sigma_2,...,\sigma_v>$ με ν στοιχεία, το στοιχείο σ_1 βρίσκεται στη **βάση της στοίβας** ενώ το σ_v στην **κορυφή**. Το πρώτο στοιχείο που τοποθετήθηκε στην στοίβα είναι το σ_1, μετά το σ_2 και ούτω καθεξής, τελευταίο το σ_v. Το πρώτο το οποίο μπορεί ν' αφαιρεθεί από την στοίβα είναι το σ_v μετά το σ_{v-1} και ούτω καθεξής, τελευταίο μπορεί να αφαιρεθεί το σ_1. Δηλαδή αυτό που τοποθετήθηκε τελευταίο αφαιρείται πρώτο ενώ αυτό που τοποθετήθηκε πρώτο αφαιρείται τελευταίο. Γι' αυτό οι στοίβες ονομάζονται **LIFO (Last In First Out)** λίστες.

8.5. Δομές Δεδομένων σε Prolog

8.5.4. Στοίβες: Εισαγωγή

- Οι πράξεις τις οποίες μπορούμε να εκτελέσουμε στη δομή της στοίβας είναι οι εξής.
  - 1) **Άδεια στοίβα** η οποία είτε δημιουργεί μια άδεια στοίβα ή ελέγχει εάν μια στοίβα είναι άδεια.
  - 2) **Λήψη στοιχείου** από την κορυφή στοίβας **(top).**
  - 3) **Διαγραφή στοιχείου** από την κορυφή της στοίβας **(pop).**
  - 4) **Καταχώρηση** ενός νέου στοιχείου στη στοίβα **(push).**
- Εφόσον η στοίβα υλοποιηθεί με λίστα, τότε οι πράξεις της διαγραφής και της καταχώρησης στοιχείου μπορούν να γίνουν **είτε** στην **αρχή** της λίστας **είτε** στο **τέλος**.
- Θα υλοποιήσουμε τις πράξεις της στοίβας και με του δύο τρόπους. Στη συνέχεια παρουσιάζουμε για κάθε πράξη της στοίβας τον περιγραφικό ορισμό της καθώς και δύο διαφορετικές υλοποιήσεις της με λίστες Prolog.

8.5. Δομές Δεδομένων σε Prolog

8.5.4. Στοίβες: : Ορισμός και υλοποίηση πράξεων στοίβας.

- ***Άδεια στοίβα***. Το κατηγόρημα **empty_stack(S:stack(T))** είναι αληθές εάν η στοίβα **S** είναι άδεια,mode(a). Αυτό το κατηγόρημα έχει μόνο μια υλοποίηση.

```
empty_stack([]).
```

Πρόγραμμα 8.76: Άδεια στοίβα.

- ***Ανάκτηση κορυφής στοίβας***. Το κατηγόρημα **top(St:stack(T), TopItem:T)** είναι αληθές εάν **TopItem** είναι το στοιχείο στη κορυφή της στοίβας **St**, mode(g,a).
  - ***Α τρόπος υλοποίησης***: Η κορυφή της στοίβας είναι στο τέλος της λίστας. **Πρόγραμμα 8.77: Κορυφή στοίβας (Α τρόπος).**

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

  - ***Β τρόπος υλοποίησης***: Η κορυφή της στοίβας είναι στην αρχή της λίστας. **Πρόγραμμα 8.78: Κορυφή στοίβας (Β τρόπος).**

```
top([H|T], H).
```

8.5. Δομές Δεδομένων σε Prolog

8.5.4. Στοίβες: : Ορισμός και υλοποίηση πράξεων στοίβας.

- ***Αφαίρεση του στοιχείου από την κορυφή της στοίβας***. Το κατηγόρημα **pop(St:stack(T), RestSt:stack(T))** είναι αληθές εάν η στοίβα **RestSt** είναι η στοίβα **St** χωρίς το στοιχείο στη κορυφής της, mode(g,a).
  - ***Α τρόπος υλοποίησης***: Η κορυφή της στοίβας είναι στο τέλος της λίστας.

```
pop([H], []).
pop([H|T], [H|Rest]) :-  pop(T, Rest).
```

Πρόγραμμα 8.79: Αφαίρεση στοιχείου από στοίβα (Α' τρόπος).

  - ***Β τρόπος υλοποίησης***: Η κορυφή της στοίβας είναι στην αρχή της λίστας.

```
pop([H|T], T).
```

Πρόγραμμα 8.80: Αφαίρεση στοιχείου από στοίβα (Β' τρόπος).

8.5. Δομές Δεδομένων σε Prolog

8.5.4. Στοίβες: : Ορισμός και υλοποίηση πράξεων στοίβας.

- ***Καταχώρηση του στοιχείου στη κορυφή της στοίβας***. Το κατηγόρημα **push(StIn:stack(T), Top:T, StOut:stack(T))** είναι αληθές εάν η στοίβα **StOut** είναι η στοίβα **StIn** με επιπλέον το στοιχείο **Top** στην κορυφή της, mode(g, g,a).
  - ***Α τρόπος υλοποίησης***: Η κορυφή της στοίβας είναι στο τέλος της λίστας.

```
push([], Elem, [Elem]).
push([H|T], Elem, [H|Rest]) :-  push(T, Elem, Rest).
```

Πρόγραμμα 8.81: Καταχώρηση στοιχείου σε στοίβα (Α' τρόπος).

  - ***Β τρόπος υλοποίησης***: Η κορυφή της στοίβας είναι στην αρχή της λίστας.

```
push(Q, X, [X|Q]).
```

Πρόγραμμα 8.82: Καταχώρηση στοιχείου σε στοίβα (Β' τρόπος).

8.5. Δομές Δεδομένων σε Prolog

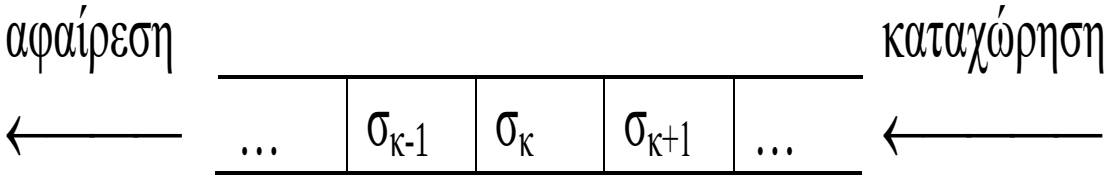
8.5.5. Ουρές: Εισαγωγή

- Μια άλλη δομή δεδομένων με μεγάλη χρήση στην ανάπτυξη εφαρμογών Τεχνητής Νοημοσύνης είναι η ***ουρά (queue)***.
- Οι ***ουρές*** είναι **ταξινομημένες λίστες** στις οποίες οι καταχωρήσεις στοιχείων γίνονται στο **πίσω μέρος της λίστας** το οποίο ονομάζεται είτε ***τέλος*** ή ***ουρά*** ενώ οι αφαιρέσεις στοιχείων γίνονται στο **εμπρόσθιο μέρος της λίστας** το οποίο ονομάζεται είτε ***αρχή*** ή ***κεφαλή***.
- Σε μια ουρά **O = <σ_1, σ_2,...,σ_{v-1}, σ_v>** με v στοιχεία το στοιχείο **σ_1** βρίσκεται πρώτο στην ουρά και θα αφαιρεθεί πρώτο ενώ το στοιχείο **σ_v** βρίσκεται τελευταίο στην ουρά και θα αφαιρεθεί τελευταίο. Δηλαδή σε μια ουρά η σειρά αφαίρεσης των στοιχείων είναι ίδια με την σειρά τοποθέτησης τους στη λίστα. Γι' αυτό οι ουρές ονομάζονται και **FIFO (First In First Out)** λίστες.
-

8.5. Δομές Δεδομένων σε Prolog

8.5.5. Ουρές: Εισαγωγή

- Το παρακάτω σχήμα μας δείχνει παραστατικά τη λειτουργία μιας ουράς.



- Οι πράξεις που μπορούμε να εκτελέσουμε στη δομή της ουράς είναι οι εξής.
  - 1) **Άδεια ουρά** η οποία είτε δημιουργεί μια άδεια ουρά ή ελέγχει εάν μια ουρά είναι άδεια.
  - 2) **Έλεγχος για ύπαρξη στοιχείου σε ουρά**.
  - 3) **Λήψη στοιχείου** από την αρχή της ουράς **χωρίς διαγραφή του**.
  - 4) **Λήψη και διαγραφή στοιχείου από την αρχή της ουράς (**dequeue).
  - 5**) Καταχώρηση** ενός νέου στοιχείου στο τέλος της ουράς (enqueue).
- Η υλοποίηση της ουράς θα γίνει με τη λίστα της Prolog. Στη συνέχεια παρουσιάζουμε για κάθε πράξη της ουράς τον περιγραφικό ορισμό της καθώς και την υλοποίηση της σε Prolog.

8.5. Δομές Δεδομένων σε Prolog

8.5.5. Ουρές: Ορισμός και υλοποίηση πράξεων ουράς.

- ***Άδεια ουρά***. Το κατηγόρημα **empty_queue(S:queue(T))** είναι αληθές εάν η ουρά **Q** είναι άδεια, mode(a).

```
empty_queue([]).
```

Πρόγραμμα 8.100: Άδεια ουρά.

- ***Έλεγχος για ύπαρξη στοιχείου σε ουρά***. Το κατηγόρημα **member_queue(X:T, Q:queue(T))** είναι αληθές εάν το στοιχείο **X** είναι στοιχείο στην ουρά **Q**, mode(a,g).

```
member_queue(X, Q) :- member(X,Q).
```

Πρόγραμμα 8.101: Έλεγχος μέλους σε ουρά.

- ***Λήψη στοιχείου από την αρχή της ουράς χωρίς να διαγραφεί από την ουρά***. Το κατηγόρημα **get_elem_queue(X:T,Q:queue(T))** είναι αληθές εάν το στοιχείο **X** είναι στη κεφαλή της ουράς **Q**, mode(a,g).

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

Πρόγραμμα 8.102: Ανάκτηση στοιχείου από την αρχή ουράς χωρίς τη διαγραφή του.

8.5. Δομές Δεδομένων σε Prolog

8.5.5. Ουρές: Ορισμός και υλοποίηση πράξεων ουράς.

- ***Λήψη και διαγραφή στοιχείου από την αρχή της ουράς*** (dequeue). Το κατηγόρημα **dequeue(X:T,Q:queue(T), NewQ:queue(T))** είναι αληθές εάν το στοιχείο **X** είναι στη κεφαλή της ουράς **Q** και **NewQ** είναι η ουρά **Q** χωρίς το στοιχείο **X**, mode(g,g,a).

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

Πρόγραμμα 8.103: Ανάκτηση στοιχείου από την αρχή ουράς με διαγραφή του.

- ***Καταχώρηση ενός νέου στοιχείου στο τέλος της ουράς*** (enqueue). Το κατηγόρημα **enqueue(X:T,Q:queue(T), NewQ:queue(T))** είναι αληθές εάν **NewQ** είναι η ουρά **Q** με το στοιχείο **X** τοποθετημένο στο τέλος της, mode(g,g,a).

```
enqueue(X, [], [X]).
enqueue(X, [H|T1], [H|T2]) :- enqueue(X, T1, T2).
```

Πρόγραμμα 8.104: Καταχώρηση στοιχείου στο τέλος ουράς.

Τέλος Διάλεξης

Ευχαριστώ!

Ερωτήσεις;