2. Δομές Δεδομένων στην PROLOG

2.1 Αναπαράσταση Δομών Δεδομένων με τη βοήθεια Ορων

Οι δομές δεδομένων στην PROLOG μπορούν να αναπαρασταθούν με τη βοήθεια του λογικού όρου (ή απλά όρου). Οπως έχουμε ήδη αναφέρει ένας όρος μπορεί να είναι:

- μία **σταθερά,**
- μία **μεταβλητή** ή
- ένας **σύνθετος (συναρτησιακός) όρος**

2.1.1. Ατομα και Αριθμοί

Οπως και στις διατακτικές γλώσσες προγραμματισμού έτσι και στην PROLOG οι σταθερες χρησιμοποιούνται για να αναπαραστήσουν τα πρωταρχικά δεδομένα της γλώσσας. Οι σταθερές μπορεί να είναι **άτομα (atoms)** ή **αριθμοί**. Ατομο είναι μία αδιαίρετη οντότητα η οποία χαρακτηρίζεται απόλυτα από το ονομά της. Αυτό σημαίνει ότι δύο άτομα που έχουν το ίδιο όνομα αναπαριστάνουν την ίδια ακριβώς οντότητα. Από συντακτική άποψη τα άτομα μπορούν να κατασκευαστούν με τρείς τρόπους:

(1) Από ένα μικρό γράμμα ακολουθούμενο από 0 ή περισσότερα γράμματα (μικρά/κεφαλαία), ψηφία ή τον ειδικό χαρακτήρα "_" (underscore):

john, greek, prolog, pROLOG, knossos,
x1, x10, x125a,
x_10, addison_wesley, computer_Science

(2) Από μία παράθεση ειδικών χαρακτήρων όπως για παράδειγμα:

-->, ..., ::=, {####}

με την επιφύλαξη ότι κάποιοι ειδικοί χαρακτήρες (**":-"**, **"?-"**) έχουν προαποφασισμένη σημασία για την PROLOG.

(3) Από μία αλυσίδα χαρακτήρων τοποθετημένη μέσα σε απλά εισαγωγικά, όπως:

'PROLOG', 'computer science', 'North_Greece', 'John'

Στις περισσότερες υλοποιήσεις της PROLOG οι αριθμοί μπορούν να είναι ακέραιοι ή πραγματικοί. Το εύρος παράστασης των αριθμών όμως διαφέρει από υλοποίηση σε υλοποίηση.

2.1.2 Μεταβλητές

Οι μεταβλητές χρησιμοποιούνται για να ονομάσουν όχι μία προκαθορισμένη οντότητα αλλά μία κατηγορία από οντότητες. Συντακτικά οι μεταβλητές κατασκευάζονται από ένα κεφαλαίο γράμμα το οποίο ακολουθείται από 0 ή περισσότερα γράμματα (κεφαλαία/μικρά), ψηφία ή τον χαρακτήτα "_' (underscore). Επίσης μία μεταβλητή μπορεί να αρχίζει με τον χαρακτήρα "_", όπως για παράδειγμα:

X, Y, Answer,
X1, X12, X1new,
New_result, _12, _x15

Κατά τη διάρκεια της εκτέλεσης μία μεταβλητή μπορεί να είναι **τοποθετημένη** (***instatiated***) ή **μή-τοποθετημένη**. Μία μεταβλητή ονομάζεται τοποθετημένη όταν υπάρχει κάποιο αντικείμενο, στο οποίο αναφέρεται και μή-τοποθετημένη όταν το αντικείμενο αυτό δεν έχει ακόμη προσδιοριστεί. Το πεδίο δράσης μιάς μεταβλητής στην PROLOG περιορίζεται μέσα στη φράση όπου αυτή εμφανίζεται. Ετσι για παράδειγμα αν θεωρήσουμε τις φράσεις :

(P) parent(X,Y) :- mother(X,Y).
(G) grandparent(X,Y) :- father(X,Z), parent(Z,Y).

η μεταβλητή **X** που εμφανίζεται στη φράση **(P)** είναι διαφορετική από τη μεταβλητή **X** που εμφανίζεται στη φράση **(G)**.

Μία μεταβλητή που αποτελείται μόνο από τον χαρακτήρα "_" (underscore) ονομάζεται ανώνυμη. Οταν μία μεταβλητή εμφανίζεται σαν παράμετρος μόνο μία φορά σε μία φράση και δεν μας ενδιαφέρει η τιμή της κατά τη διάρκεια της εκτέλεσης, μπορούμε στη θέση της να χρησιμοποιήσουμε την ανώνυμη μεταβλητή, όπως για παράδειγμα στον παρακάτω κανόνα:

employee(X) :- employs(_,X).

όπου δηλώνουμε ότι ο **X** είναι υπάλληλος έαν υπάρχει κάποιος (ο _) που τον έχει προσλάβει

2.1.3. Σύνθετες Δομές Δεδομένων

Οι σύνθετες οντότητες μπορούν να δομηθούν με τη βοήθεια συναρτησιακών όρων. Για παράδειγμα ο σύνθετος όρος:

point(X, Y, Z)

μπορεί να χρησιμοποιηθεί για την αναπαράσταση ενός σημείου στο χώρο. Σύνθετοι όροι μπορούν να χρησιμοποιηθούν σαν ορίσματα σε άλλους σύνθετους όρους για να δημιουργήσουν ακόμα πιο σύνθετες δομές. Για παράδειγμα ο όρος :

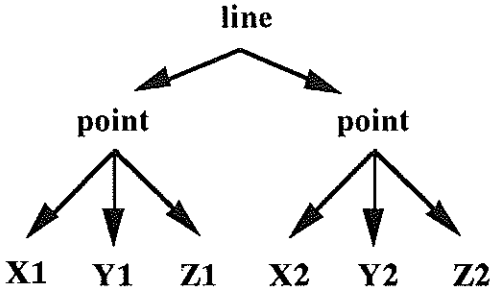
line(point(X1, Y1, Z1), point(X2, Y2, Z2))

μπορεί να χρησιμοποιηθεί για την αναπαράσταση της ευθείας που περνά από τα σημεία **point(X1, Y1, Z1)** και **point(X2, Y2, Z2)**.

Οι όροι μπορούν να παρασταθουν γενικά με τη βοήθεια δέντρων. Στη ρίζα του δέντρου αντιστοιχίζεται το συναρτησιακό σύμβολο του όρου και στα υποδεντρα αντιστοιχίζονται τα ορίσματα του. Για παράδειγμα στον όρο:

line(point(X1,Y1,Z1), point(X2,Y2,Z2))

αντιστοιχεί το δέντρο:



Πολλές φορές είναι πιο κατάλληλο να γράψουμε έναν όρο χρησιμοποιώντας **ενθεματική** ***(infix)***, ή **επιθεματική** ***(postfix)*** σημειογραφία αντί της **προθεματικής** ***(prefix)***, όπως **a+b** ή **(a,b)+** αντί **+(a,b)**. Ολες οι υλοποιήσεις της PROLOG διαθέτουν ένα μηχανισμό για τη δήλωση ενός συναρτησιακού συμβόλου σαν προθεματικού, ενθεματικού ή επιθεματικού, καθώς και για τη δήλωση προτεραιοτήτων ανάμεσα σε συναρτησιακά σύμβολα. Γιά παράδειγμα δηλώνοντας τα σύμβολα **&** και **=>** σαν ενθεματικά με το πρώτο υψηλότερης προτεραιότητας από το δεύτερο, ο όρος:

=>(&(p,q), &(s,r))

μπορεί να γραφεί πιο ευανάγνωστα σαν:

p&q => s&r

Μια ειδική περίπτωση όρου που χρησιμοποιείται ιδιαίτερα είναι η λίστα. Η λίστα με στοιχεία τα **a, b, c, d** μπορεί να παρασταθεί από τον όρο **.(a,.(b,.(c,.(d,nil))))**. Μια πιο ευανάγνωστη μορφή αναπαράστασης της λίστας αυτής που έχει υιοθετηθεί από τις περισσότερες υλοποιήσεις της PROLOG είναι η "**[a, b, c, d]**", και προκύπτει αν χρησιμοποιηθεί το κόμμα "," στη θέση της τελείας "." και δηλωθουν τα συναρτησιακά σύμβολα "," , "[", και "]" σαν ενθεματικό, προθεματικό και επιθεματικό αντίστοιχα, με κατάλληλες προτεραιότητες μεταξύ τους. Επίσης ο όρος **[A|B]** παριστάνει μια λίστα με πρώτο στοιχείο το **A** και ουρά το **B**. (Το σύμβολο "|" έχει οριστεί σαν ενθεματικό). Στις λίστες θά αναφερθούμε με περισσότερη λεπτομέρεια σε επόμενη παράγραφο.

Θα πρέπει να τονίσουμε ότι εκτός από τον συντακτικό τρόπο γραφής, καμμία άλλη πληροφορία (όπως για παράδειγμα δηλώσεις τύπων) δεν είναι αναγκαία στην PROLOG για να αναγνωριστεί το είδος μιάς δομής.

2.2 Αναπαράσταση Δομών Δεδομένων με τη βοήθεια Φράσεων

Φράσεις μπορούν να χρησιμοποιηθούν για την αναπαράσταση δομών δεδομένων, όπως πίνακες, λίστες, δέντρα κ.λπ. που αλλιώς παριστάνονται με τη βοήθεια όρων. Στην περίπτωση αυτή, για να ορίσουμε κάποια δομή δεδομένων, χρησιμοποιούμε ένα ή περισσότερα κατηγορήματα. Για παράδειγμα αντί να αναπαραστήσουμε τη λίστα με στοιχεία **a,b,c,d** με τον όρο **[a,b,c,d]** μπορούμε να της δώσουμε ένα όνομα, έστω **list1**, και να την αναπαραστήσουμε με τις φράσεις:

```
item(list1,1,a).
item(list1,2,b).
item(list1,3,c).
item(list1,4,d).
length(list1,4).
```

όπου το κατηγόρημα **item(X,Y,Z)** ερμηνεύεται σαν "**Z** είναι το **Y**-οστό στοιχείο της λίστας **X**" και το κατηγόρημα **length(X,Y)** ερμηνεύεται σαν "το μήκος της λίστας **X** είναι **Y**". Η χρήση των φράσεων αποδεικνύεται ιδιαίτερα αποτελεσματική για την αναπαράσταση ειδικού τύπου δομών. Για παράδειγμα ο μοναδιαίος πίνακας

$$\begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{pmatrix}$$

μπορεί να παρασταθεί απλά από τις φράσεις:

```
array(X,X,1).
array(X,Y,0) :- X =/= Y.
```

όπου το κατηγόρημα **array(X,Y,Z)** ερμηνεύεται σαν "**Z** είναι το στοιχείο του πίνακα που βρίσκεται στη **X** γραμμή και στη **Y** στήλη". Παρόλο που στην PROLOG, όπως προαναφέραμε. δεν υπάρχει τυπικά διαφορά ανάμεσα στα συναρτησιακά σύμβολα και στα ονόματα των κατηγορημάτων, η χρήση κατηγορημάτων όταν χρησιμοποιούμε φράσεις αντί όρων για την αναπαράσταση δομών δεδομένων παρουσιάζει ποιοτική διαφορά. Αυτό κυρίως γιατί στην περίπτωση που χρησιμοποιούμε κατηγορήματα οι δομές δεδομένων που αναπαριστάνουμε γίνονται άμεσα εκτελέσιμες.

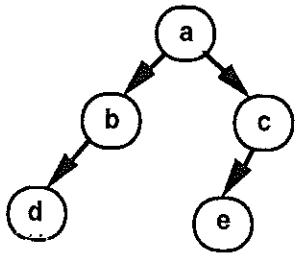
2.3 Ασκήσεις

1] Πώς μπορούμε να αναπαρασταθεί στην PROLOG με τη βοήθεια ενός συναρτησιακού όρου:

(α) Ενα τρίγωνο
(β) Ενα τετράγωνο
(γ) Ενα κύκλο

Με βάση την αναπαράσταση αυτή να γραφεί ένα κατηγόρημα που να ελέγχει έαν ένας όρος αυτός αντιστοιχεί στο σχήμα της κάθε περίπτωσης.

2] Θεωρήστε την παρκάτω αναπαράσταση ενός δυαδικού δέντρου:
(α) Το κενό δέντρο αναπαριστάνεται με τη σταθερά nil.
(β)Το μή κενό δέντρο αναπαριστάνεται με τη βοήθεια του συναρτησιακού όρου **btree(Node, Left, Right)**, όπου **Node** είναι η τιμή της ρίζας του δέντρου και **Left** και **Right** είναι το αριστερό και το δεξιό υποδέντρο αντίστοιχα, που παριστάνονται ανδρομικά σαν δυαδικά δέντρα. Για παράδειγμα ο συναρτησιακός όρος που αντιστοιχεί στο παρακάτω δυαδικό δέντρο:



είναι ο:

btree(a,btree(b,btree(d,nil,nil),nil),btree(c,btree(e,nil,nil),nil))

Να γραφεί το κατηγόρημα **is_btree(T)**, το οποίο να ελέγχει εάν ο όρος **T** είναι δυαδικό δέντρο και το κατηγόρημα **btree_member(X,T)**, το οποίο να ελέγχει εάν το **X** υπάρχει στο δέντρο **T**.