Ασκήσεις Φυλλάδιο 6

Κατηγορήματα Συλλογής Λύσεων και η Βάση Γνώσης της Prolog/Τελεστές

1. Έστω τα ακόλουθα γεγονότα τα οποία αποτελούν κάποιου είδους μετρήσεις από μια βιομηχανική μονάδα:

sample(2).
sample(5).
sample(14).
sample(7).
sample(26).

Υλοποιείστε ένα Prolog κατηγόρημα **less_than_ten/1** το οποίο επιστρέφει στο όρισμά του πόσες από αυτές έχουν τιμή μικρότερη του 10. Για παράδειγμα:

?-less_than_ten(X).
X=3

2. Να ορίσετε ένα κατηγόρημα **set_diff_f/3** το οποίο υλοποιεί αφαίρεση συνόλων στα δύο πρώτα του κατηγορήματα τα οποία είναι λίστες. Να χρησιμοποιηθεί το κατηγόρημα **findall/3**. Για παράδειγμα:

?- set_diff_f([1, 2, 3, 4], [1, 2], L).
L = [3, 4]

?- set_diff_f([1, 2, 3, 4], [1, 2, 5, 6], L).
L = [3, 4]

?- set_diff_f([a, b], [1, 6, 3], L).
L = [a, b]

3. Χρησιμοποιώντας το κατηγόρημα **setof/3** να ορίσετε το κατηγόρημα **minlist(Min,List)**, που επιτυγχάνει όταν το **Min** είναι το ελάχιστο στοιχείο της λίστας **List**. Για παράδειγμα:

?- minlist(X, [1, 3, 2, 4, 5, 12, 15, 3, 0]).
X = 0

?- minlist(X, [3, 2, 4, 5, 12, 15]).
X = 2

4. Να ορίσετε ένα κατηγόρημα **proper_set_s(List),** το οποίο επιτυγχάνει όταν η λίστα **List** περιέχει μόνο μοναδικά στοιχεία, είναι δηλαδή σύνολο. Να χρησιμοποιήσετε τον ορισμό το **setof/3.** Για παράδειγμα:

?- proper_set_s([a, b, c, d, d, e, f]).
No
?- proper_set_s([a, b, c, d, e, f]).
Yes

5. Να ορίσετε το κατηγόρημα **map_f(Operation,List,Results)** το οποίο δεδομένου ενός ονόματος κατηγορήματος τάξης 2 **Operation,** και μιας λίστας **List,** επιστρέφει τη λίστα **Results** που περιέχει τα αποτελέσματα της εφαρμογής του κατηγορήματος στη δοθείσα λίστα. Για την υλοποίηση να χρησιμοποιήσετε το κατηγόρημα **findall/3**:

?- map_f(double, [1, 2, 3, 4], L).
L = [2, 4, 6, 8]
Yes
?- map_f(square, [1, 2, 3, 4], L).
L = [1, 4, 9, 16]
Yes

Οι ορισμοί των **double(X,Y)** και **square(X,Y)** θα δοθούν από εσάς.

?- double(3,X)
X = 6
?- square(3,X)
X = 9

6. Υλοποιήστε δύο Prolog κατηγορήματα **push/1** και **pop/1** που υλοποιούν μια κλασική στοίβα σαν καθολική μεταβλητή. Τα περιεχόμενα της στοίβας θα αποθηκευτούν ***σε ένα δυναμικό κατηγόρημα*** **stack/1** στη βάση της Prolog. Για παράδειγμα:

stack([]). % μια άδεια στοίβα
stack([12,34,5,6,7]). % μη-άδεια στοίβα
Αρχικά υπάρχει η κενή στοίβα. Η χρήση των κατηγορημάτων φαίνεται στην παρακάτω ακολουθία ερωτημάτων της Prolog:

?- push(a).
Yes
?- push(b).
Yes
?- push(c).
Yes
?- pop(X).
X = c
Yes
?- pop(X).
X = b
Yes
?- pop(X).
X = a
Yes
?- pop(X).
No

7. Να επεκτείνετε το πρόγραμμα που αποτιμά **εκφράσεις προτασιακής λογικής**, που βρίσκεται στις διαφάνειες με τους ακόλουθους τελεστές

- άρνηση (--). Ο μοναδιαίος τελεστής της άρνησης έχει μεγαλύτερη προτεραιότητα (παίρνει ορίσματα πρώτος) από τους τελεστές and και or, καθώς και έχει δεξιά προσεταριστικότητα.
- συνεπαγωγή (==>). Ο τελεστής της συνεπαγωγής μπορεί να υλοποιηθεί από το ισοδύναμό

του **a==>b <==> -- a or b**. Ο τελεστής έχει μικρότερη προτεραιότητα από τους τελεστές **and/or**.

- αποκλειστική διάζευξη (**xor**), ίδιας προτεραιότητας και προσεταιριστικότητας με τον **or**.
- τον τελεστή **nor,** ίδιας προτεραιότητας και προσεταιριστικότητας με τον **or.** και
- τον τελεστή **nand**, ίδιας προτεραιότητας και προσεταιριστικότητας με τον **and**.

Το πρόγραμμα θα πρέπει να μπορεί να αποτιμά εκφράσεις της μορφής

?- t and t.
Yes

?- t and t or f.
Yes

?- t and -- t.
No

?- t and t or f xor t nand f.
no
Μπορούμε να χρησιμοποιήσουμε μεταβλητές στις θέσεις των t και f? Γιατί?

8. α) Να αναπτύξετε ένα κατηγόρημα **model/1**, το οποίο δέχεται μια έκφραση της προτασιακής λογικής, όπως εκείνες που φαίνονται στην παραπάνω άσκηση (άσκηση 7) το οποίο επιτρέπει την ύπαρξη μεταβλητών στις εκφράσεις.

Για την υλοποίηση θα χρειαστείτε το κατηγόρημα της ECliPSe Prolog (*ήδη υλοποιημένο*) **term_variables/2** (**term_variables(Term,Vars)**) το οποίο επιστρέφει στη λίστα Vars, όλες τις μεταβλητές που εμφανίζονται στον όρο Term. Παράδειγμα:

?- term_variables(test(X, 23, in(Y)), Vars).
X = X
Y = Y
Vars = [Y, X]
Yes (0.00s cpu)

?- term_variables(X and t, Vars).
X = X
Vars = [X]
Yes (0.00s cpu)

Παράδειγμα εκτέλεσης του κατηγορήματος **model/1**:

?- model(t and X).
X = t
Yes (more)
No

?- model(Y and X).
Y = t
X = t
Yes (more)
No

?- model(Y or X).
Y = t
X = t
Yes (more)
Y = t
X = t
Yes (more)
Y = f
X = t
Yes (more)
Y = t
X = f
Yes (more)
No (0.03s cpu)

β) Να αναπτύξετε ένα κατηγόρημα **theory/1** το οποίο δέχεται μια λίστα από εκφράσεις προτασιακής λογικής, που πιθανά να περιέχουν μεταβλητές, και αποτιμά τις εκφράσεις αυτές (σύξευξη). Για παράδειγμα:

?- theory([Y, Y xor X, X and Z, Y ==> Z]).
No

?- theory([Y or X, X and Z, Y nand Z]).
Y = f
X = t
Z = t
Yes (more)
No

γ) "Αν το **Y** είναι αληθές, τότε **Y <==> X or Y**, για οποιαδήποτε Χ". Η διπλή συνεπαγωγή μπορεί να γραφεί σαν σύζευξη των **Y ==> X or Y** και **X or Y ==> Y.** Χρησιμοποιώντας την **setof/3** και το κατηγόρημα **theory/1** μπορείτε να αποδείξετε το παραπάνω; (Δεν απαιτείται να ανεβάσετε την λύση).

9. Έστω το Prolog κατηγόρημα **seperate_lists(List,Lets,Nums)** το οποίο δοθείσας μιας λίστας ακεραίων και χαρακτήρων (ατόμων) **List**, πετυχαίνει όταν στην λίστα **Lets** είναι όλοι οι χαρακτήρες (άτομα) της **List** και στην λίστα **Nums**, όλοι οι αριθμοί. Για παράδειγμα:

?- seperate_lists([1, 2, a, b, c, 3, 4, 5, f], Lets, Nums).
Lets = [a, b, c, f]
Nums = [1, 2, 3, 4, 5]
Yes (0.00s cpu)
?- seperate_lists([1, 2], Lets, Nums).
Lets = []
Nums = [1, 2]
Yes (0.00s cpu)
?- seperate_lists([a, b], Lets, Nums).
Lets = [a, b]

Nums = []
Yes (0.00s cpu)

Να δώσετε τον **ΜΗ-αναδρομικό ορισμό** του κατηγορήματος χρησιμοποιώντας κατηγορήματα συλλογής λύσεων.

(Σημ: Το κατηγόρημα **number(N)** πετυχαίνει όταν το Ν είναι αριθμός).