ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ

Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών

Λογικός Προγραμματισμός

Ασκήσεις

Παναγιώτης Σταματόπουλος

Τμήμα Πληροφορικής και Τηλεπικοινωνιών

Περιεχόμενα

1. Α’ Ομάδα Ασκήσεων "Λογικού Προγραμματισμού" Ακαδημαϊκού Έτους 2014-15

1.1 Άσκηση 1

Δέκα «φίλοι», ο Antti, ο Eric, ο Harris, ο Jeroen, ο Johan, ο Luis, ο Michel, ο Peter, ο Wolfgang και ο Yanis, έχουν καθίσει γύρω από ένα κυκλικό τραπέζι για να συζητήσουν. Γνωρίζουμε ότι ισχύουν τα εξής:

1. Ο Antti κάθεται επτά θέσεις δεξιά από τον Johan.
2. Ο Johan κάθεται εννέα θέσεις δεξιά από τον Luis.
3. Ο Jeroen κάθεται επτά θέσεις δεξιά από τον Eric.
4. Ο Jeroen κάθεται έξι θέσεις δεξιά από τον Michel.
5. Ο Eric κάθεται οκτώ θέσεις δεξιά από τον Harris.
6. Ο Wolfgang κάθεται οκτώ θέσεις δεξιά από τον Antti.
7. Ο Wolfgang κάθεται επτά θέσεις δεξιά από τον Peter.

Ορίστε σε Prolog ένα κατηγόρημα `seats/1` έτσι ώστε το `seats(Seats)` να επιστρέφει στο Seats τη λίστα των φίλων με τη σειρά, από αριστερά προς τα δεξιά, που κάθονται στο τραπέζι. Ενδεικτική εκτέλεση:

```
?- seats(Seats).

Seats = [yanis,antti,peter,jeroen,johan,luis,eric,michel,

         harris,wolfgang]
```

Να σημειωθεί ότι το πρόβλημα αυτό έχει ακριβώς μία λύση, αν εξαιρέσουμε τις συμμετρικές της παραπάνω, που προκύπτουν από αυτήν μέσω περιστροφής. Το κατηγόρημα που θα ορίσετε δεν θα πρέπει να επιστρέφει συμμετρικές λύσεις ως εναλλακτικές.

Επεκτείνετε το πρόγραμμά σας ορίζοντας ένα κατηγόρημα `seats/2` έτσι ώστε το `seats(Rules,Seats)` να επιστρέφει στο Seats τη λίστα των φίλων, θεωρώντας ότι ισχύουν οι κανόνες της λίστας Rules (αύξοντες αριθμοί), όπως φαίνονται παραπάνω. Όταν υπάρχουν περισσότερες από μία λύση, θα πρέπει να επιστρέφονται όλες μέσω οπισθοδρόμησης. Ενδεικτικές εκτελέσεις:

```
?- seats([1,2,3,4,5,6,7],Seats).

Seats = [yanis,antti,peter,jeroen,johan,luis,eric,michel,

         harris,wolfgang]                                --> ;

No


?- seats([1,2,3,4,5,6],Seats).
```

```
Seats = [yanis,jeroen,johan,luis,eric,michel,harris,wolfgang,
         peter,antti]                                    --> ;
Seats = [yanis,antti,peter,jeroen,johan,luis,eric,michel,
         harris,wolfgang]                                --> ;
No

?- seats([2,3,4,5,6,7],Seats).
Seats = [yanis,johan,luis,wolfgang,jeroen,antti,peter,eric,
         michel,harris]                                  --> ;
Seats = [yanis,antti,peter,jeroen,johan,luis,eric,michel,
         harris,wolfgang]                                --> ;
Seats = [yanis,wolfgang,jeroen,antti,peter,eric,michel,harris,
         johan,luis]                                     --> ;
No

?- seats([1,2,4,5,7],Seats).
Seats = [yanis,michel,johan,luis,eric,wolfgang,harris,jeroen,
         peter,antti]                                    --> ;
Seats = [yanis,jeroen,johan,luis,wolfgang,michel,eric,peter,
         harris,antti]                                   --> ;
Seats = [yanis,antti,michel,wolfgang,johan,luis,peter,eric,
         jeroen,harris]                                  --> ;
Seats = [yanis,antti,peter,jeroen,johan,luis,eric,michel,
         harris,wolfgang]                                --> ;
Seats = [yanis,harris,wolfgang,antti,jeroen,peter,johan,luis,
         michel,eric]                                    --> ;
Seats = [yanis,peter,eric,antti,harris,jeroen,johan,luis,
         wolfgang,michel]                                --> ;
Seats = [yanis,michel,eric,wolfgang,harris,antti,peter,jeroen,
         johan,luis]                                     --> ;
```

```
Seats = [yanis,eric,jeroen,harris,wolfgang,antti,michel,peter,
         johan,luis]                                    --> ;
No


?- findall(Seats,seats([],Seats),L),length(L,N),
   write('Found '),write(N),write(' solutions'),nl,fail.
Found 362880 solutions
No (0.87s cpu)


?- findall(Seats,seats([1,4,7],Seats),L),length(L,N),
   write('Found '),write(N),write(' solutions'),nl,fail.
Found 480 solutions
No (0.00s cpu)


?- findall(Seats,seats([2,6],Seats),L),length(L,N),
   write('Found '),write(N),write(' solutions'),nl,fail.
Found 4320 solutions
No (0.05s cpu)
```

Στην υλοποίηση των δύο κατηγορημάτων **δεν επιτρέπεται να χρησιμοποιήσετε απολύτως κανένα ενσωματωμένο κατηγόρημα της Prolog**. Οτιδήποτε χρειάζεστε που τυχαίνει να είναι ενσωματωμένο κατηγόρημα και είναι δυνατόν να οριστεί σε «καθαρή» Prolog (π.χ. κατηγορήματα διαχείρισης λιστών), θα πρέπει να το ορίσετε μέσα στο αρχείο που θα παραδώσετε, με όνομα της δικής σας επιλογής, όχι αυτό που έχει ως ενσωματωμένο.

Παραδοτέο για την άσκηση είναι **ένα πηγαίο αρχείο Prolog** με όνομα **seats.pl**, μέσα στο οποίο θα πρέπει να είναι ορισμένα και τα δύο κατηγορήματα που ζητούνται.

Bonus ερώτηση 1 (10%): Με 10 άτομα στο τραπέζι και 7 περιορισμούς απόστασης, είχαμε μοναδική λύση. Υπήρχε περίπτωση για 10 άτομα και λιγότερους από 7 περιορισμούς, να υπήρχε πάλι μόνο μία λύση; Αν ναι, ποιοι θα πρέπει να ήταν οι περιορισμοί αυτοί;

Bonus ερώτηση 2 (20%): Αν τα άτομα στο τραπέζι ήταν 20, ποιοι πιστεύετε θα έπρεπε να είναι οι περιορισμοί απόστασης ελαχίστου πλήθους, ώστε να υπάρχει μοναδική λύση στο πρόβλημα; Εφαρμόστε την πρότασή σας για τα άτομα που βρίσκονται στον πίνακα: http://en.wikipedia.org/wiki/Eurogroup#Members.

Οι απαντήσεις σας στις bonus ερωτήσεις θα πρέπει να ενσωματωθούν μέσα στο αρχείο **seats.pl** που θα παραδώσετε, είτε με τη μορφή σχολίων, είτε με τη μορφή υλοποίησης κατηγορημάτων, με διαφορετικά ονόματα από αυτά που ζητούνται στο βασικό τμήμα της άσκησης.

1.2 Άσκηση 2

Υλοποιήστε σε Prolog ένα κατηγόρημα `diags(Matrix,DiagsDown,DiagsUp)`, το οποίο να παίρνει σαν όρισμα ένα πίνακα `Matrix`, στη μορφή μίας λίστας λιστών (οι εσωτερικές λίστες είναι οι γραμμές του πίνακα), και να επιστρέφει στα `DiagsDown` και `DiagsUp` τις λίστες των στοιχείων των κατιουσών και των ανιουσών διαγωνίων του πίνακα, αντίστοιχα. Ένα παράδειγμα εκτέλεσης είναι το εξής:

```
?- diags([[a,b,c,d],[e,f,g,h],[i,j,k,l]],DiagsDown,DiagsUp).

DiagsDown = [[i],[e,j],[a,f,k],[b,g,l],[c,h],[d]]

DiagsUp = [[a],[b,e],[c,f,i],[d,g,j],[h,k],[l]]
```

Παραδοτέο για την άσκηση είναι **ένα πηγαίο αρχείο Prolog** με όνομα **diags.pl**.

1.3 Άσκηση 3

Έστω ότι έχετε στη διάθεσή σας έναν επεξεργαστή που διαθέτει N καταχωρητές R_1, R_2, ..., R_N οι οποίοι έχουν δακτυλιοειδή διευθέτηση. Αυτό σημαίνει ότι μπορεί κάποιος να μεταφέρει τα περιεχόμενα του καταχωρητή R_i στον καταχωρητή R_{i+1}, για $1 \leq i < N$, και του R_N στον R_1 με τις εντολές `move(i)`, για $1 \leq i < N$, και `move(N)`, αντίστοιχα. Επίσης, μπορεί να αντιμεταθέσει τα περιεχόμενα των καταχωρητών R_i και R_j με την εντολή `swap(i,j)`, όπου $i < j$. Έστω, τώρα, ότι σας δίνονται τα αρχικά περιεχόμενα των N καταχωρητών, καθώς και τα επιθυμητά τελικά περιεχόμενα. Το ζητούμενο είναι να βρεθεί η μικρότερη αλληλουχία από τις εντολές `move` και `swap` που πρέπει να εκτελεσθούν για να επιτευχθεί ο ζητούμενος μετασχηματισμός. Συγκεκριμένα, ορίστε το κατηγόρημα `codegen/3`, έτσι ώστε όταν αυτό καλείται με πρώτο όρισμα τη λίστα των αρχικών περιεχομένων των καταχωρητών και με δεύτερο όρισμα τη λίστα των τελικών περιεχομένων, να επιστρέφει στο τρίτο όρισμα τη λίστα των απαραίτητων (ελάχιστων) εντολών που απαιτούνται για το μετασχηματισμό. Σημειώστε ότι είναι δυνατόν στην αναπαράσταση των περιεχομένων των καταχωρητών να έχουμε, τόσο στην αρχική όσο και στην τελική κατάσταση, το σύμβολο `*`, που σημαίνει, για μεν την αρχική κατάσταση "δεν ξέρω τι περιέχεται στον καταχωρητή", για δε την τελική κατάσταση "δεν με ενδιαφέρει τι περιέχεται στον καταχωρητή". Κάποια παραδείγματα εκτέλεσης είναι τα εξής:

```
?- codegen([a,b,c,d],[a,d,a,b],L).

L = [move(2),move(1),swap(3,4),swap(2,3)]


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

L = [move(1),move(3)]


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

L = [move(1)]
```

```
?- codegen([a,b,c,d,e,f],[f,f,b,e,a,e],L).

L = [move(2),swap(4,6),move(5),swap(4,5),swap(1,5),move(1)]
```

Παραδοτέο για την άσκηση είναι **ένα πηγαίο αρχείο Prolog** με όνομα **codegen.pl**.

Σημειώματα

Σημείωμα Ιστορικού Εκδόσεων Έργου

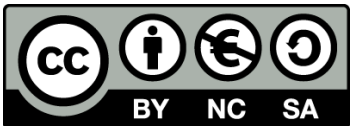
Το παρόν έργο αποτελεί την έκδοση 1.0.

Σημείωμα Αναφοράς

Copyright Εθνικόν και Καποδιστριακόν Πανεπιστήμιον Αθηνών, Παναγιώτης Σταματόπουλος. «Λογικός Προγραμματισμός, Η γλώσσα προγραμματισμού Prolog». Έκδοση: 1.0. Αθήνα 2015. Διαθέσιμο από τη δικτυακή διεύθυνση: http://opencourses.uoa.gr/courses/DI117/.

Σημείωμα Αδειοδότησης

Το παρόν υλικό διατίθεται με τους όρους της άδειας χρήσης Creative Commons Αναφορά, Μη Εμπορική Χρήση Παρόμοια Διανομή 4.0 [1] ή μεταγενέστερη, Διεθνής Έκδοση. Εξαιρούνται τα αυτοτελή έργα τρίτων π.χ. φωτογραφίες, διαγράμματα κ.λ.π., τα οποία εμπεριέχονται σε αυτό και τα οποία αναφέρονται μαζί με τους όρους χρήσης τους στο «Σημείωμα Χρήσης Έργων Τρίτων».



[1] http://creativecommons.org/licenses/by-nc-sa/4.0/

Ως **Μη Εμπορική** ορίζεται η χρήση:

- που δεν περιλαμβάνει άμεσο ή έμμεσο οικονομικό όφελος από την χρήση του έργου, για το διανομέα του έργου και αδειοδόχο
- που δεν περιλαμβάνει οικονομική συναλλαγή ως προϋπόθεση για τη χρήση ή πρόσβαση στο έργο
- που δεν προσπορίζει στο διανομέα του έργου και αδειοδόχο έμμεσο οικονομικό όφελος (π.χ. διαφημίσεις) από την προβολή του έργου σε διαδικτυακό τόπο

Ο δικαιούχος μπορεί να παρέχει στον αδειοδόχο ξεχωριστή άδεια να χρησιμοποιεί το έργο για εμπορική χρήση, εφόσον αυτό του ζητηθεί.

Διατήρηση Σημειωμάτων

- Οποιαδήποτε αναπαραγωγή ή διασκευή του υλικού θα πρέπει να συμπεριλαμβάνει:
- το Σημείωμα Αναφοράς
- το Σημείωμα Αδειοδότησης
- τη δήλωση Διατήρησης Σημειωμάτων

- το Σημείωμα Χρήσης Έργων Τρίτων (εφόσον υπάρχει)

μαζί με τους συνοδευόμενους υπερσυνδέσμους.

Σημείωμα Χρήσης Έργων Τρίτων

Το Έργο αυτό κάνει χρήση των ακόλουθων έργων:

Εικόνες/Σχήματα/Διαγράμματα/Φωτογραφίες

Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στο πλαίσιο του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «**Ανοικτά Ακαδημαϊκά Μαθήματα στο Πανεπιστήμιο Αθηνών**» έχει χρηματοδοτήσει μόνο τη αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.