Λογικός Προγραμματισμός με Περιορισμούς

Τμ. Εφαρμοσμένης Πληροφορικής

Εργασία 1 2020-2021

1. List Processing (35/100)

Έστω το Prolog κατηγόρημα **extend(List, NewList)** το οποίο επιτυγχάνει όταν δοθείσας μιας λίστας η οποία περιέχει όρους της μορφής **(N,X)**, όπου N ένας από τους αριθμούς **0,1,2** και X ένας όρος, η λίστα **NewList** περιέχει λίστες N όρων X. Για παράδειγμα:

```
?- extend([(1, a), (2, b)], L).
L = [[a], [b, b]]
?- extend([(0, a), (0, b), (1, c)], L).
L = [[c]]
?- extend([(1, a), (2, b), (0, a), (2, c)], L).
L = [[a], [b, b], [c, c]]
```

Να δώσετε τον ορισμό του κατηγορήματος.

2. Flights (45/100)

Οι πληροφορίες πτήσεων σε μια αεροπορική εταιρεία είναι αποθηκευμένες σε γεγονότα της ακόλουθης μορφής:

```
flight(Code,fromto(Dep,Dest),etd(Time),eta(Time),cost(Cost)).
```

όπου τον πρώτο όρισμα είναι ο κωδικός της πτήσης, το δεύτερο (**`fromto(Dep,Dest)`**), είναι ένας σύνθετος όρος με το αεροδρόμιο αναχώρησης (**`Dep`**) και το αεροδρόμιο άφιξης (**`Dest`**), το τρίτο όρισμα ο χρόνος αναχώρησης της πτήσης (estimated time of departure - **`etd`**), το τέταρτο ο χρόνος άφιξης της πτήσης (estimated time of arrival – **`eta`**), και το τελευταίο το κόστος της πτήσης. Για παράδειγμα το ακόλουθο γεγονός:

```
flight(oa123,fromto(skg,ath),etd(10),eta(11),cost(120)).
```

δηλώνει ότι η πτήση με κωδικό **`oa123`**, θα αναχωρήσει από Θεσσαλονίκη (**`skg`**) για Αθήνα (**`ath`**) στις 10 το πρωί (**`etd(10)`**), θα φτάσει στην Αθήνα στις 11 (**`eta(11)`**), και κοστίζει 120 ευρώ (**`cost(120)`**).

Συνήθως, δεν υπάρχουν απευθείας πτήσεις ανάμεσα σε προορισμούς. Στην περίπτωση αυτή, προσφέρεται στον πελάτη ένας συνδυασμός πτήσεων, με ενδιάμεσους σταθμούς (αεροδρόμια), αρκεί η άφιξη της προηγούμενης πτήσης να είναι τουλάχιστον 1 ώρα από την αναχώρηση της επόμενης (=<) και να μην επισκέπτεται το ίδιο αεροδρόμιο δύο φορές. Ο συνδυασμός πρέπει να είναι μέσα στην ίδια μέρα.

(α) Να υλοποιήσετε ένα Prolog κατηγόρημα **`find_flight(Dept,Dest,Plan,Cost,ETA)`** (**`find_flight/5`**), το οποίο πετυχαίνει, όταν η λίστα **`Plan`** περιέχει μια έγκυρη ακολουθία πτήσεων από τον τόπο αναχώρησης **`Dept`**, στον προορισμό **`Dest`**, με συνολικό κόστος **`Cost`**, και αναμενόμενο χρόνο άφιξης **`ETA`**. Η ώρα αναχώρησης μπορεί να είναι οποιαδήποτε. Προφανώς το κατηγόρημα θα επιστέφει όλες τις πιθανές ακολουθίες πτήσεων. Για παράδειγμα:

```
?- find_flight(skg, ath, Plan, Cost, ETA).
Plan = [oa123]
Cost = 120
ETA = 11
```

```
Yes (0.00s cpu, solution 1, maybe more)
Plan = [oa124]
Cost = 80
ETA = 13
Yes (0.00s cpu, solution 2, maybe more)
Plan = [oa125]
Cost = 40
ETA = 14
Yes (0.00s cpu, solution 3, maybe more)
Plan = [oa120, bt100]
Cost = 170
ETA = 19
Yes (0.00s cpu, solution 4, maybe more)
No

?- find_flight(skg, fra, Plan, Cost, ETA).
Plan = [oa123, lf200]
Cost = 670
ETA = 18
Yes (0.00s cpu, solution 1, maybe more)
Plan = [oa124, lf200]
Cost = 630
ETA = 18
Yes (0.00s cpu, solution 2, maybe more)
Plan = [oa125, lf200]
Cost = 590
ETA = 18
Yes (0.00s cpu, solution 3, maybe more)
No (0.00s cpu)

?- find_flight(skg, edi, Plan, Cost, ETA).
Plan = [oa123, aa120, bt110]
Cost = 270
ETA = 23
Yes (0.00s cpu, solution 1, maybe more)
(...Υπάρχουν και άλλες λύσεις στο ερώτημα).
```

(β) Να υλοποιήσετε ένα κατηγόρημα **waiting_time(Flights,WTime)** το οποίο πετυχαίνει αν ο συνολικός χρόνος αναμονής ανάμεσα στις διαδοχικές πτήσεις της λίστας **Flights** είναι **WTime**. Ο χρόνος αναμονής είναι ο χρόνος αναχώρησης της επόμενης πτήσης μείον τον χρόνο άφιξης της προηγούμενης. Για παράδειγμα:

```
?- waiting_time([oa123], WTime).
WTime = 0
Yes (0.00s cpu)

?- waiting_time([oa120, bt100], WTime).
WTime = 4
Yes (0.00s cpu)

?- waiting_time([oa123, aa120, bt110], WTime).
WTime = 7
Yes (0.00s cpu)
```

(γ) Να υλοποιήσετε ένα κατηγόρημα **select_flight/5**:

select_flight(Dep,Dest,Plan,Before,Cost,MinWait),

το οποίο πετυχαίνει όταν η λίστα **Plan**, περιέχει την ακολουθία πτήσεων από το **Dep** στον προορισμό **Dest**, η οποία φτάνει στο **Dest** πριν (=<) από τον χρόνο **Before**, έχει κόστος **Cost** και έχει τον ελάχιστο χρόνο αναμονής **MinWait** από όλους τους συνδυασμούς πτήσεων. Αν δεν υπάρχει τέτοιος συνδυασμός, το κατηγόρημα αποτυγχάνει. Για παράδειγμα:

```
?- select_flight(skg, fra, Plan, 24, Cost, MinWait).
Plan = [oa125, lf200]
Cost = 590
MinWait = 1
Yes (0.00s cpu)

?- select_flight(skg, fra, Plan, 12, Cost, MinWait).
No (0.00s cpu)
```

3. Reductions (20/100)

Να υλοποιήσετε ένα κατηγόρημα **reduction(List,Val)** (**reduction/2**) το οποίο πετυχαίνει αν η **Val** είναι ο ακέραιος στον οποίο αποτιμάται η αριθμητική έκφραση σε ανάστροφη Πολωνική γραφή (reverse Polish notation) που περιέχεται στην λίστα **List**. Η έκφραση **List** μπορεί να περιέχει τους δυαδικούς (διμελής) τελεστές **+,-,*,//,min,max** ή τον μοναδιαίο τελεστή **abs**. Αν η έκφραση δεν έχει σωστή σύνταξη τότε το κατηγόρημα αποτυγχάνει. Αν δεν χρησιμοποιήσετε την συνήθη προσέγγιση με στοίβα, κερδίζετε extra μονάδες. Παραδείγματα εκτέλεσης:

```
?- reduction([4, 2, +], Val).
Val = 6
Yes (0.00s cpu, solution 1, maybe more)

?- reduction([4, 2, -], Val).
Val = 2
Yes (0.00s cpu, solution 1, maybe more)

?- reduction([2, 4, -], Val).
Val = -2
Yes (0.00s cpu, solution 1, maybe more)

?- reduction([2], Val).
Val = 2
Yes (0.00s cpu, solution 1, maybe more)

?- reduction([2, 4, max, 3, +], Val).
Val = 7
Yes (0.00s cpu, solution 1, maybe more)

?- reduction([2, 4, max, -3, abs, +], Val).
Val = 7
Yes (0.00s cpu, solution 1, maybe more)
```

```
?- reduction([2, 4, min, 3, abs, *], Val).
Val = 6
Yes (0.00s cpu, solution 1, maybe more)

?- reduction([2, 4, min, *], Val).
No (0.00s cpu)
```

ΠΑΡΑΔΟΣΗ

Θα παραδώσετε εντός της ημερομηνίας που αναφέρεται στο ECLASS τα ακόλουθα:

- Ένα αρχείο με το όνομα **exec1.ecl** το οποίο θα περιέχει τις λύσεις (κατηγορήματα) **και των τριών ασκήσεων.**
- Το απαραίτητο αρχείο βρίσκονται στο ECLASS στην ενότητα **Έγγραφα -> Coursework**
- Ένα αρχείο **report.pdf** (***σε μορφή pdf***) το οποίο θα περιέχει:
  - **Στην πρώτη σελίδα το Όνομά σας, τον Αριθμό μητρώου σας και το email σας.**
  - Για κάθε μια από τις τρείς ασκήσεις:
    - τον **κώδικα** (ασχέτως αν βρίσκεται και στο αρχείο exec1.ecl) και σχολιασμό σχετικά με αυτόν.
    - Παραδείγματα εκτέλεσης (2 για κάθε κατηγόρημα)
    - Bugs και προβλήματα που έχει ο κώδικάς σας.

ΠΡΟΣΟΧΗ: ΝΑ ΑΚΟΛΟΥΘΗΣΕΤΕ ΑΥΣΤΗΡΑ ΤΑ ΟΝΟΜΑΤΑ ΚΑΙ ΤΗ ΣΕΙΡΑ ΤΩΝ ΟΡΙΣΜΑΤΩΝ ΠΟΥ ΔΙΝΕΤΑΙ ΠΑΡΑΠΑΝΩ (**ΑΥΤΟΜΑΤΟΣ ΕΛΕΓΧΟΣ ΚΩΔΙΚΑ**)

Καλή επιτυχία (*και have fun with Prolog!*)