Μεταπτυχιακό Πρόγραμμα Εφαρμοσμένης Πληροφορικής

Προχωρημένη Τεχνητή Νοημοσύνη

Εργασία 1

Prolog-Λογικός Προγραμματισμός και αναζήτηση
Χειμερινό Εξάμηνο 2016 - 2017

Άσκηση 1 (The Company)

Σας δίνεται το αρχείο **company.ecl** το οποίο περιέχει γεγονότα με πληροφορίες για τους υπαλλήλους μιας εταιρείας, ως γεγονότα. Τα ακόλουθα είναι μερικά παραδείγματα για να γίνει κατανοητή η πληροφορία που υπάρχει στα αντίστοιχα γεγονότα του αρχείου.
Στοιχεία εργαζόμενου:

```
employee(name(john),occupation([programmer,analyst]),wage(40000)).
```

- Δηλώνεται ότι ο υπάλληλος john, εργάζεται ως προγρραμματιστής και αναλυτής (programmer, analyst) με ετήσιο μισθό 40000 ευρώ.

```
employee(name(alice),occupation([programmer,tester]),wage(35000)).
```

- Δηλώνει ότι η υπάλληλος alice, εργάζεται ως προγραμματίστρια και στην δοκιμή λογισμικού, με ετήσιο μισθό 35000 ευρώ.

Προσωπικά στοιχεία εργαζόμενου:

```
data(john,status(married,children(2))).
```

- Δηλώνει ότι ο john είναι παντρεμένος και έχει 2 παιδιά.

```
data(alice,status(single,children(0))).
```

- Δηλώνει ότι η alice δεν είναι παντρεμένη και δεν έχει παιδία.

α) Να υλοποιήσετε ένα κατηγόρημα **wage/2 (wage(Empl,Wage)**, το οποίο πετυχαίνει όταν ο Empl είναι υπάλληλος τη εταιρείας με ετήσιο μισθό Wage. Κατά την οπισθοδρόμηση θα επιστρέφονται όλες οι λύσεις. Για παράδειγμα:

```
?- wage(Empl, Wage).
Empl = john
Wage = 40000
Yes (0.00s cpu, solution 1, maybe more)

Empl = alice
Wage = 35000
Yes (0.00s cpu, solution 2, maybe more)

Empl = peter
Wage = 25000
Yes (0.01s cpu, solution 3, maybe more)
... (υπάρχουν και άλλες λύσεις)
```

β) Να υλοποιήσετε ένα κατηγόρημα **single_with_children(Empl,N)**, το οποίο πετυχαίνει αν ο υπάλληλος Empl δεν είναι παντρεμένος(single) και έχει N παιδιά, όπου N > 0. Για παράδειγμα:

```
?- single_with_children(Empl, N).
Empl = helen
N = 2
```

```
Yes (0.00s cpu, solution 1, maybe more)
Empl = donald
N = 1
Yes (0.00s cpu, solution 2, maybe more)
No (0.00s cpu)
```

γ) Να υλοποιήσετε ένα κατηγόρημα **same_status/2 (same_status(X,Y))**, το οποίο πετυχαίνει όταν οι υπάλληλοι X και Y είναι της ίδιας οικογενειακής κατάστασης (married / single). Για παράδειγμα:

```
?- same_status(X, Y).
X = john
Y = peter
Yes (0.00s cpu, solution 1, maybe more)
X = john
Y = nick
Yes (0.00s cpu, solution 2, maybe more)
X = john
Y = mathiew
Yes (0.00s cpu, solution 3, maybe more)
X = john
Y = igor
Yes (0.01s cpu, solution 4, maybe more)
... (και άλλες λύσεις)
```

δ) Να υλοποιήσετε ένα κατηγόρημα **benefit(Name,Wage,Benefit)** το οποίο πετυχαίνει όταν ο υπάλληλος Name, με ετήσιο μισθό Wage, παίρνει επίδομα Benefit. Το επίδομα καθορίζεται ως εξής:

- Αν ο υπάλληλος δεν είναι παντρεμένος και δεν έχει παιδιά το επίδομα είναι 0.
- Αν ο υπάλληλος δεν είναι παντρεμένος και έχει παιδιά, το επίδομα είναι 1000 + 500 για κάθε παιδί.
- Αν ο υπάλληλος είναι παντρεμένος το επίδομα είναι 500 + 600 ευρώ για κάθε παιδί (αν έχει).

Για παράδειγμα:

```
?- benefit(john, Wage, Benefit).
Wage = 40000
Benefit = 1700
Yes

?- benefit(helen, Wage, Benefit).
Wage = 140000
Benefit = 2000
Yes (0.00s cpu, solution 1, maybe more)

?- benefit(alice, Wage, Benefit).
Wage = 35000
Benefit = 0
Yes (0.00s cpu, solution 1, maybe more)

?- benefit(Name, Wage, 0).
Name = alice
Wage = 35000
Yes (0.00s cpu, solution 1, maybe more)

Name = bob
Wage = 15000
Yes (0.01s cpu, solution 2, maybe more)
```

```
No (0.01s cpu)

?- benefit(Name, Wage, Benefit).
Name = alice
Wage = 35000
Benefit = 0
Yes (0.00s cpu, solution 1, maybe more)

Name = bob
Wage = 15000
Benefit = 0
Yes (0.00s cpu, solution 2, maybe more)

Name = helen
Wage = 140000
Benefit = 2000
Yes (0.01s cpu, solution 3, maybe more)
```

Άσκηση 2 (Αναδρομικές Συναρτήσεις)

Έστω η ακόλουθη αναδρομική συνάρτηση:

$$f(n,k)=\begin{cases} n*k & if\ n=k \\ n*f(n-1,k) & if\ n>k \\ k*f(n,k-1) & if\ n<k \end{cases}$$

Ορίστε ένα Prolog κατηγόρημα fn/3 που να υλοποιεί την παραπάνω συνάρτηση. Το κατηγόρημα θα πρέπει να έχει την ακόλουθη συμπεριφορά:

```
?- fn(3,3,R).
R = 9
Yes
?- fn(3,4,R).
R = 36
Yes
?- fn(4,3,R).
R = 36
Yes
?- fn(5,3,R).
R = 180
Yes
```

Άσκηση 3 (Google Alien Language)

Μετά από χρόνια ερευνών, οι άνθρωποι της google ανακάλυψαν ότι οι (α) υπάρχουν εξωγήινοι και (β) στέλνουν μηνύματα που αποτελούνται από γράμματα. Λόγω θορύβου, σε συγκεκριμένες θέσεις των μηνυμάτων που λαμβάνονται δεν είναι σίγουρο ποιο γράμμα αντιστοιχεί, αλλά υπάρχει ένα σύνολο γραμμάτων τα οποία μπορεί να είναι έγκυρα. Έτσι, κάθε λέξη που έχει ληφθεί δίνεται ως πρότυπο (pattern) με την μορφή λίστας. Στο πρότυπο (λίστα), σε κάθε θέση μπορεί να υπάρχει:

- μόνο ένας χαρακτήρας, τότε η τελική λέξη περιέχει το συγκεκριμένο αυτό χαρακτήρα στη αντίστοιχη θέση,
- μια λίστα χαρακτήρων, οπότε στην τελική λέξη μπορεί να υπάρχει *μόνο ένας από αυτούς τους χαρακτήρες* στην αντίστοιχη θέση.

Παραδείγματα:

Πρότυπο	Πιθανές Λέξεις
[a,b,c]	[a,b,c]
[a,**[b,c]**,d]	[a,**b**,d] *και* [a,**c**,d]
[**[a,b,c]**,e,f]	[**a**,e,f], [**b**,e,f] *και* [**c**,e,f]

α) Να δημιουργήσετε ένα κατηγόρημα **generate_word(Pattern,Word)**, το οποίο δοθέντος ενός προτύπου **Pattern** ενοποιεί το δεύτερο όρισμά του **Word** με μια πιθανή λέξη. Κατά την οπισθοδρόμηση θα πρέπει να παράγονται όλες οι πιθανές λέξεις. Για παράδειγμα:

```
?- generate_word([a, b, c], W).
W = [a, b, c]
Yes (0.00s cpu)

?- generate_word([a, [b, c], d], W).
W = [a, b, d]
Yes
W = [a, c, d]
Yes  (2 λύσεις)
```

β) Να γράψετε ένα κατηγόρημα **count_possible_words(Pattern,Count)** το οποίο δοθέντος ενός προτύπου **Pattern** ενοποιεί την μεταβλητή **Count** με τον αριθμό των πιθανών λέξεων που αντιστοιχούν στο **Pattern**. Για παράδειγμα:

```
?- count_words([a, b, c], Count).
Count = 1
Yes
?- count_words([a, [b, c], d], Count).
Count = 2
Yes
?- count_words([[a, b, c], e, f], Count).
Count = 3
Yes
?- count_words([[a, b, c], [e, f], g, [h, i, j]], Count).
Count = 18
Yes
```

Υποδείξεις :

- Στο ερώτημα (α) να χρησιμοποιήσετε αναδρομή!
- Θα σας φανεί χρήσιμο το κατηγόρημα **is_list/1** το οποίο επιτυγχάνει όταν το όρισμά του είναι λίστα. Για παράδειγμα:

```
?- is_list([d, e, f]).
Yes
?- is_list(a).
No
```

Άσκηση 4 (Λίστες)

Έστω το Prolog κατηγόρημα **split_list(Limit,List,High,Low)** (**split_list/4**) το οποίο δοθείσας μιας λίστας ακεραίων **List**, και ενός ακεραίου **Limit**, "επιστρέφει" στην λίστα **High** όλα τα στοιχεία της λίστας List που είναι *μεγαλύτερα ή ίσα* του **Limit**, και στη λίστα Low τα στοιχεία που είναι μικρότερα του **Limit.** Για παράδειγμα:

```
?- split_list(2, [0, 2, 1, 3, 4], High, Low).
High = [2, 3, 4]
Low = [0, 1]
```

```
?- split_list(0, [0, 2, 1, 3, 4], High, Low).
High = [0, 2, 1, 3, 4]
Low = []
Yes

?- split_list(4, [0, 2, 1, 3, 4], High, Low).
High = [4]
Low = [0, 2, 1, 3]
Yes
```

α) Να δώσετε τον **αναδρομικό ορισμό** του κατηγορήματος κάνοντας όσο το **δυνατό λιγότερους** ελέγχους.

β) Να ορίσετε ένα κατηγόρημα **split_list_alt/3** με την ίδια δηλωτική σημασία με το προηγούμενο της άσκησης 4.(a) **μη αναδρομικό ορισμό**, βασιζόμενοι σε κάποιο κατηγόρημα συλλογής λύσεων. (**Σημ**: **χρησιμοποιώντας κατηγορήματα συλλογής λύσεων** (όχι αναδρομή). Για παράδειγμα:

```
?- split_list_alt(0, [-1, 0, 2, 1, 3, 4], High, Low).
High = [0, 2, 1, 3, 4]
Low = [-1]
Yes (0.00s cpu)

?- split_list_alt(3, [-1, 0, 2, 1, 3, 4], High, Low).
High = [3, 4]
Low = [-1, 0, 2, 1]
Yes (0.00s cpu)
```

Άσκηση 5 (Το πρόβλημα των πολλαπλών προμηθευτών)

Μια εταιρεία έχει ένα πλήθος προμηθευτών από τους οποίους αγοράζει προϊόντα που χρειάζεται. Κάθε μέρα η εταιρεία ορίζει το σύνολο των προϊόντων που χρειάζεται και ζητά από τους προμηθευτές της για προσφορές. Δυστυχώς, στη συνηθισμένη περίπτωση κανένας από τους προμηθευτές δεν μπορεί να προσφέρει όλα τα προϊόντα και άρα κάθε προμηθευτής κάνει προσφορές για ένα υποσύνολο αυτών. Άρα η εταιρεία πρέπει να επιλέξει πολλαπλούς προμηθευτές μέσα στην μέρα. Όμως,

- Οι προμηθευτές κάνουν είτε **απλές** (για ένα προϊόν) είτε **σύνθετες** προσφορές (για πολλά προϊόντα). Οι σύνθετες προσφορές είναι συνήθως συμφέρουσες, για παράδειγμα ο προμηθευτής 4 προσφέρει το αγαθό 1 για 11 μονάδες τιμής, το αγαθό 4 για 48 και τα δύο για 54 μονάδες τιμής.
- Η εταιρεία μπορεί να πάρει μόνο μια προσφορά από τον κάθε προμηθευτή.
- Δεν απαιτείται να αγοραστούν προϊόντα από κάθε προμηθευτή.
- Η εταιρεία πρέπει να πάρει όλα τα προϊόντα που χρειάζεται.

Το πρόβλημα αναπαρίσταται σαν ένα σύνολο γεγονότων. Για παράδειγμα οι προσφορές φαίνονται σαν Prolog γεγονότα της ακόλουθης μορφής, όπου το πρώτο όρισμα είναι ο προμηθευτής και το δεύτερο η λίστα με τις προσφορές του. Κάθε προσφορά αναπαρίσταται με το (Items, Price) όπου το Items είναι η λίστα με τα προϊόντα που προσφέρει και Price η τιμή. Για παράδειγμα ο provider(1) προσφέρει το προϊόν 5 για 23 μονάδες τιμής και τα προϊόντα 5,1,4 και 3 για 47 μονάδες.

offers(provider(1), [**([5], 23)**, ([1], 23), ([2], 43), ([4], 29), ([3], 14), **([5, 1, 4, 3], 47)**]).
offers(provider(2), [([3], 32), ([5], 29), ([4], 37), ([1], 17), ([6], 32), ([5, 4, 1], 72)]).
offers(provider(3), [([2], 41), ([3], 32), ([1], 29), ([5], 22), ([6], 46), ([2, 3, 1], 46), ([5, 6], 34)]).
offers(provider(4), [([2], 32), ***([1], 11)***, ([3], 49), ***([4], 48)***, ([5], 44), ([3, 5], 70), ***([1, 4], 54)***]).

Η λίστα των προιόντων είναι επίσης ένα γεγονός της μορφής

goods_to_buy([1, 2, 3, 4, 5, 6]).

Μια πιθανή λύση σε αυτό το πρόβλημα θα ήταν η ακόλουθη:

Provider	Προσφορά
provider(1)	*([5], 23)*
provider(2)	*([6], 32)*
provider(3)	*([2, 3, 1], 46)*
provider(4)	*([4], 48)*

Να υλοποιήσετε ένα κατηγόρημα **solve/2** το οποίο επιλύει το πρόβλημα και επιστρέφει το συνολικό κόστος, που είναι το άθροισμα των προσφορών που έχουν επιλεγεί.

```
?- solve(Solution,Cost)
Solution = [(provider(4),[4],48), (provider(3),[2,3,1],46),
(provider(2),[6],32), (provider(1),[5],23)]
Cost = 149
```

Extra Marks: Μπορείτε να βρείτε μαι ευρετική συνάρτηση και να την εφαρμόσετε στο πρόβλημα? (φυσικά να δώσετε και μια υλοποίηση).

ΠΑΡΑΔΟΣΗ

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

- Ένα αρχείο **company.ecl** που περιέχει τη λύση της **άσκησης 1.**
- Ένα αρχείο με το όνομα **exec24.ecl** το οποίο θα περιέχει τις λύσεις (κατηγορήματα) **των ασκήσεων 2-4.**
- Ένα αρχείο **multiple_providers.ecl** με τη λύση της άσκησης 5.
- 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 ασκήσεις:
    - τον **κώδικα** (ασχέτως αν βρίσκεται και στα αρχεία) και σχολιασμό σχετικά με αυτόν.
    - Παραδείγματα εκτέλεσης (1-2 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!*)