Σύνθετες δομές - Ταυτοποίηση

- Να ορισθούν κατάλληλα γεγονότα, τα οποία να αναπαριστούν τους υπαλλήλους μιας εταιρίας, με τα ακόλουθα στοιχεία.
- Κάθε ένα από τα στοιχεία δεν είναι απλά, αλλά έχουν δική τους εσωτερική δομή, δηλαδή αποτελούνται από άλλα στοιχεία.

  - Αύξων Αριθμός Υπαλλήλου
  - Ονοματεπώνυμο
    - ✓ Όνομα
    - ✓ Επώνυμο
    - ✓ Πατρώνυμο
  - Ταυτότητα
    - ✓ Αριθμός ταυτότητας
      - Γράμμα
      - Αριθμός
    - ✓ Ημερομηνία έκδοσης
      - Ημέρα
      - Μήνας
      - Έτος
    - ✓ Εκδούσα αρχή
      - Αστυνομικό τμήμα
      - Πόλη
  - Διεύθυνση
    - ✓ Οδός
    - ✓ Αριθμός
    - ✓ Πόλη
  - Ημερομηνία προσλήψεως
    - ✓ Ημέρα
    - ✓ Μήνας
    - ✓ Έτος
  - Τμήμα επιχείρησης

Γεγονότα υπαλλήλων

```
employee(
 4444,
 data(
   name(
       first(nick),
       last(bassiliades),
       middle(dimitrios)),
   id(
       number(
           letter(n),
           no(355324)),
       authority(
           at(a),
           city(thessaloniki)),
       date(23,6,1983)),
   address(
       road(tsimiski),
       number(4),
       city(thessaloniki)),
   hire_date(
       date(10,10,1999)),
   department(
       informatics)
 )
).
```

```
number(n,355324)
```

```
at(a,thessaloniki)
```

```
date(
    day(10),
    month(10),
    year(1999))
```

Κανόνες για υπαλλήλους

❖ Να ορισθούν τα ακόλουθα κατηγορήματα, τα οποία να χρησιμοποιούν τα γεγονότα των υπαλλήλων που έχουν οριστεί προηγουμένως.

- **lives_in_now(Employee,City)**
  - Ο υπάλληλος **Employee** (αύξων αριθμός) ζει στην πόλη **City**
- **works_on(Employee,Dept)**
  - Ο υπάλληλος **Employee** εργάζεται στο τμήμα **Dept**
- **same_dept(Employee1,Employee2)**
  - Οι δύο υπάλληλοι εργάζονται στο ίδιο τμήμα
- **same_dept(Employee1,Employee2,Dept)**
  - Οι δύο υπάλληλοι εργάζονται στο ίδιο τμήμα **Dept**
- **same_name(Employee1,Employee2)**
  - Οι δύο υπάλληλοι έχουν το ίδιο όνομα
- **same_road(Employee1,Employee2)**
  - Οι δύο υπάλληλοι μένουν στον ίδιο δρόμο
- **same_house(Employee1,Employee2)**
  - Οι δύο υπάλληλοι μένουν στο ίδιο σπίτι
- **hired_same_month(Employee1,Employee2)**
  - Οι δύο υπάλληλοι προσελήφθησαν τον ίδιο μήνα
- **brother(Employee1,Employee2)**
  - Οι δύο υπάλληλοι είναι αδέλφια
- **lives_in_permanently(Employee,City)**
  - Ο υπάλληλος **Employee** μένει "μόνιμα" στην πόλη **City**

Κανόνες για υπαλλήλους

```
lives_in_now(Employee,City) :-
   employee(Employee,
        data(_,_,address(_,_,city(City)),_,_)).
```

❖ Η κλήση του παραπάνω κανόνα γίνεται ως εξής:

```
?- lives_in_now(4444,C).
   C = thessaloniki
?- lives_in_now(E,thessaloniki).
   E = 4444
?- lives_in_now(E,C).
  E = 4444,   C = thessaloniki
```

❖ Εάν δεν ξέρουμε τον αύξοντα αριθμό του υπαλλήλου και θέλουμε να τον αναζητήσουμε με το όνομά του πρέπει να αλλάξουμε τον ορισμό.

```
lives_in_now2(Name,City) :-
   employee(_,
           data(Name,_,
                  address(_,_,city(City)),_,_)).
```

❖ Όταν καλούμε το κατηγόρημα θα πρέπει να δώσουμε το όνομα με τη μορφή που είναι αποθηκευμένο.

```
?- lives_in_now2(
        name(
            first(nick),
            last(bassiliades),
            middle(dimitrios)),
        C).
  C = thessaloniki
```

Κανόνες για υπαλλήλους

❖ Αν δεν ξέρουμε όλα τα στοιχεία του ονόματος μπορούμε να δώσουμε μερικά.

```
?- lives_in_now2(
       name(
           first(nick),
           last(bassiliades),
           middle(M)),
       C).
   C = thessaloniki, M = dimitrios

?- lives_in_now2(
       name(
           first(nick),
           last(bassiliades),
           middle(_)),
       C).
  C = thessaloniki
```

❖ Αν υπάρχουν περισσότεροι υπάλληλοι με συνωνυμία τότε θα πάρουμε περισσότερες λύσεις.

```
?- lives_in_now2(
       name(
           first(nick),
           last(L),
           middle(_)),
       C).
   L = bassiliades,  C = thessaloniki;
   L = nikolaidis,       C = serres;
   ...
```

Κανόνες για υπαλλήλους

❖ Αν δε θέλουμε να γράφουμε πολύπλοκη κλήση, τότε μπορούμε να αντικαταστήσουμε τη δομή με περισσότερα ορίσματα.

```
lives_in_now3(First,Last,Middle,City) :-
    employee(_,
        data(
            name(
                first(First),
                last(Last),
                middle(Middle)),
            _,
            address(_,_,city(City)),_,_)).
```

❖ Οι κλήσεις τώρα γίνονται με πιο απλό τρόπο.

```
?- lives_in_now3(nick,bassiliades,dimitrios,C).
   C = thessaloniki
?- lives_in_now3(nick,bassiliades,M,C).
   C = thessaloniki, M = dimitrios
?- lives_in_now3(nick,bassiliades,_, C).
   C = thessaloniki
?- lives_in_now3(nick,L,_,C).
   L = bassiliades,  C = thessaloniki;
   L = nikolaidis,       C = serres;
   ...
```

Κανόνες για υπαλλήλους

```
works_on(E,Dept) :-
   employee(E,data(_,_,_,_,department(Dept))).
```

όχι

```
works_on(E,Dept) :-
   employee(E,data(_,_,_,_,Dept)).
```

❖ Γιατί τότε αντί να επιστρέψει το όνομα του τμήματος, π.χ. `informatics`, επιστρέφει το σύνθετο όρο `dept(informatics)`

```
same_dept(E1,E2) :-
   employee(E1,data(_,_,_,_,department(Dept))),
   employee(E2,data(_,_,_,_,department(Dept))).
```

ή

```
same_dept(E1,E2) :-
   employee(E1,data(_,_,_,_,X)),
   employee(E2,data(_,_,_,_,X)).
```

❖ Γιατί μας ενδιαφέρει η ομοιότητα και όχι το συγκεκριμένο τμήμα

ή

```
same_dept(E1,E2) :-
   works_on(E1,Dept),
   works_on(E2,Dept).
```

Κανόνες για υπαλλήλους

```
same_name(E1,E2) :-
   employee(E1,data(name(first(Name),_,_),
                                    _,_,_,_)),
   employee(E2,data(name(first(Name),_,_),
                                    _,_,_,_)).

same_road(E1,E2) :-
   employee(E1,data(_,_,address(road(R),_,_),
                                        _,_)),
   employee(E2,data(_,_,address(road(R),_,_),
                                        _,_)).

same_house(E1,E2) :-
   employee(E1,data(_,_,address(R,N,C),_,_)),
   employee(E2,data(_,_,address(R,N,C),_,_)).

hired_same_month(E1,E2) :-
   employee(E1,data(_,_,_,hire_date(_,M,Y),_)),
   employee(E2,data(_,_,_,hire_date(_,M,Y),_)).

brother(E1,E2) :-
   employee(E1,data(name(_,Last,Middle),
                                    _,_,_,_)),
   employee(E2,data(name(_,Last,Middle),
                                    _,_,_,_)).

lives_in_permanently(E,C) :-
   employee(E,
       data(_,id(_,authority(_,city(C)),_),
              address(_,_,city(C)),_,_)).
```

❖ Όταν κάποιος εξακολουθεί να ζει στην πόλη που έβγαλε ταυτότητα, τότε έχει στην πόλη αυτή μόνιμη κατοικία (αυθαίρετο!)
  ➢ Έχουμε **τριπλή** ισότητα.

Περισσότεροι Κανόνες για υπαλλήλους

❖ Να γραφεί κατηγόρημα **transferred/3**, το οποίο να επιτυγχάνει όταν ένας υπάλληλος έχει μεταφερθεί από ένα τμήμα **A** μιας εταιρίας σε ένα άλλο **B**.

```
transferred(E,A,B) :-
   employee(E,data(_,_,_,HD1,department(A))),
   employee(E,data(_,_,_,HD2,department(B))),
   is_before(HD1,HD2).
```

❖ Υποθέτουμε ότι κάποιος υπάλληλος υπάρχει 2 φορές στη βάση δεδομένων, με διαφορετικό τμήμα και ημερομηνία πρόσληψης.
❖ Αν η μία ημερομηνία πρόσληψης είναι κατοπινή της άλλης τότε έγινε μεταφορά του υπαλλήλου από το ένα τμήμα στο άλλο.

```
is_before(hire_date(D1,M1,Y1),
          hire_date(D2,M2,Y2)) :-
   Y2 > Y1.
is_before(hire_date(D1,M1,Y),
          hire_date(D2,M2,Y)) :-
   M2 > M1.
is_before(hire_date(D1,M,Y),
          hire_date(D2,M,Y)) :-
   D2 > D1.
```

Άλλη χρονιά.

Ίδια χρονιά. Άλλο μήνα

Ίδια χρονιά και μήνα. Άλλη μέρα.

❖ Όταν δεν μας ενδιαφέρει η τιμή κάποιας μεταβλητής μπορεί να αντικατασταθεί με ανώνυμο μεταβλητή.

```
is_before(hire_date(_,_,Y1),hire_date(_,_,Y2)) :-
   Y2 > Y1.
is_before(hire_date(_,M1,Y),hire_date(_,M2,Y)) :-
   M2 > M1.
is_before(hire_date(D1,M,Y),hire_date(D2,M,Y)) :-
   D2 > D1.
```

Περισσότεροι Κανόνες για υπαλλήλους

❖ Να γραφεί κατηγόρημα **temp_transferred/2**, το οποίο να επιτυγχάνει όταν ένας υπάλληλος έχει μεταφερθεί προσωρινά από ένα τμήμα **A** μιας εταιρίας σε κάποιο άλλο και μετά από κάποιο χρονικό διάστημα ξαναμεταφέρθηκε πίσω.

```
temp_transferred(E,A) :-
   transferred(E,A,B),
   transferred(E,B,A).
```

❖ Εάν ο υπάλληλος έχει περάσει από πολλά ενδιάμεσα τμήματα μέχρι να ξαναγυρίσει στο αρχικό του τμήμα, τότε το παραπάνω κατηγόρημα θα βρει τη σωστή απάντηση?

```
employee(1234,data(...,hire_date(9,5,84),
                       dept(informatics))).
employee(1234,data(...,hire_date(10,10,88),
                       dept(accounting))).
employee(1234,data(...,hire_date(27,1,90),
                       dept(warehouse))).
employee(1234,data(...,hire_date(2,3,90),
                       dept(informatics))).
```

Σύνδεση Υπαλλήλου με Τμήμα

❖ Έστω ότι υπάρχουν γεγονότα **department**, τα οποία έχουν διάφορες πληροφορίες για τα τμήματα της εταιρίας.

```
department(name(informatics),
           location(thessaloniki),
           employees(24)).
```

❖ Εάν ο υπάλληλος ζει σε διαφορετική πόλη από το τμήμα στο οποίο δουλεύει τότε πρέπει να μεταφερθεί.

```
must_be_transferred(E) :-
   employee(E,data(_,_,address(_,_,C),_,
                           dept(D))),
   department(name(D),location(C1),_),
   C1 \= C.
```

❖ Ένας υπάλληλος μπορεί να μεταφερθεί σε τμήμα που βρίσκεται στην πόλη που μένει.

```
can_be_transferred_to(E,D) :-
   must_be_transferred(E),
   employee(E,data(_,_,address(_,_,C),_,
                           dept(D))),
   department(name(D),location(C),_).
```

❖ Στον προηγούμενο ορισμό υπάρχει διπλή κλήση (?) του γεγονότος **employee** (Είναι σωστό αλλά όχι αποδοτικό).

```
can_be_transferred_to(E,D) :-
   employee(E,data(_,_,address(_,_,C),_,
                           dept(D1))),
   department(name(D1),location(C1),_),
   C1 \= C,
   department(name(D),location(C),_).
```

Φυσικοί Αριθμοί ως Σύνθετοι Όροι

❖ Ένας φυσικός αριθμός μπορεί να αναπαρασταθεί ως σύνθετος όρος ως εξής:
- Το μηδέν (0) αναπαρίσταται ως απλός όρος: **0**
- Το ένα (1) αναπαρίσταται ως ο σύνθετος όρος: **s(0)**
  - ✓ Σημασία: *Το 1 είναι ο επόμενος αριθμός από το μηδέν*
- Το δύο (2) αναπαρίσταται ως ο σύνθετος όρος: **s(s(0))**
- **...**

❖ Να γραφεί κατηγόρημα **natural_number/1** το οποίο να πετυχαίνει όταν το όρισμά του είναι φυσικός αριθμός.

```
natural_number(0).
natural_number(s(X)) :-
   natural_number(X).
```

❖ Το παραπάνω κατηγόρημα ελέγχει αν κάποιος αριθμός είναι φυσικός, αλλά και μπορεί να παράγει όλους τους φυσικούς αριθμούς.

```
?- natural_number(0).
yes
?natural_number(s(s(0))) :-
yes
?-natural_number(N).
N = 0 ;
N = s(0) ;
N = s(s(0)) ;
...
```

Φυσικοί Αριθμοί ως Σύνθετοι Όροι

❖ Να γραφεί κατηγόρημα **plus/3** το οποίο να προσθέτει δύο φυσικούς αριθμούς και να επιστρέφει το άθροισμά τους στο τρίτο όρισμα.

```
plus(0,X,X).
plus(s(X),Y,s(Z)) :- plus(X,Y,Z).
```

❖ Το παραπάνω κατηγόρημα έχει πολλαπλές λειτουργίες:

➢ Ελέγχει αν κάποιος αριθμός είναι το άθροισμα άλλων δύο

```
?- plus(s(0),s(s(0)),s(s(s(0)))).
yes
?- plus(s(0),s(s(0)),s(s(s(s(0))))).
no
```

➢ Υπολογίζει το άθροισμα δύο φυσικών αριθμών

```
?- plus(s(0),s(s(0)),C).
C = s(s(s(0)))
```

➢ Υπολογίζει τη διαφορά δύο φυσικών αριθμών!

```
?- plus(s(0),B,s(s(s(0)))).
B = s(s(0))
?- plus(A,s(s(0)),s(s(s(0)))).
A = s(0)
```

➢ Βρίσκει όλα τα ζεύγη αριθμών που αν προστεθούν μπορούν να μας δώσουν έναν φυσικό αριθμό

```
?- plus(A,B,s(s(s(0)))).
A = 0 , B = s(s(s(0))) ;
A = s(0) , B = s(s(0)) ;
A = s(s(0)) , B = s(0) ;
A = s(s(s(0))) , B = 0 ;
no
```

Φυσικοί Αριθμοί ως Σύνθετοι Όροι

❖ Να γραφεί κατηγόρημα **times/3** το οποίο να πολλαπλασιάζει δύο φυσικούς αριθμούς και να επιστρέφει το γινόμενό τους στο τρίτο όρισμα.

➢ Εκμεταλλευόμαστε την ύπαρξη της άθροισης.

```
times(0,Y,0).
times(s(X),Y,Z1) :-
   times(X,Y,Z),
   plus(Z,Y,Z1).
```

❖ Το παραπάνω κατηγόρημα έχει πολλαπλές λειτουργίες:

```
?- times(s(s(0)),s(s(s(0))),C).
C = s(s(s(s(s(0)))))

?- times(s(s(0)),B,s(s(s(s(s(s(0)))))).
B = s(s(s(0)))

?- times(A,s(s(0)),s(s(s(s(s(s(0)))))).
A = s(s(s(0)))

?- times(A,B,s(s(s(s(s(s(0)))))).
A = s(0) , B = s(s(s(s(s(s(0)))))) ;
A = s(s(0)) , B = s(s(s(0))) ;
A = s(s(s(0))) , B = s(s(0)) ;
A = s(s(s(s(s(0))))) , B = s(0)
```

Φυσικοί Αριθμοί ως Σύνθετοι Όροι

❖ Με τον ίδιο τρόπο μπορούμε να ορίσουμε την ύψωση σε δύναμη.

```
power(X,0,s(0)).
power(X,s(Y),Z1) :-
   power(X,Y,Z),
   times(Z,X,Z1).
```

❖ Το παραπάνω κατηγόρημα έχει πολλαπλές λειτουργίες:

```
?- power(s(s(0)),s(s(0)),C).
C = s(s(s(s(0))))

?- power(A,s(s(s(0))),s(s(s(s(s(s(s(s(s(0)))))))))).
A = s(s(0))

?- power(s(s(s(0))),B,s(s(s(s(s(s(s(s(s(0)))))))))).
B = s(s(0))

?- power(A,B,s(s(s(s(s(s(s(s(s(0)))))))))).
A = s(s(s(s(s(s(s(s(s(0)))))))))  , B = s(0) ;
A = s(s(s(0)))  , B = s(s(0))
```

Φυσικοί Αριθμοί ως Σύνθετοι Όροι

❖ Σύστημα 2 εξισώσεων με 2 αγνώστους.

```
% A1*X + B1*Y = C1
% A2*X + B2*Y = C2
system(equation(A1,B1,C1),equation(A2,B2,C2),X,Y) :-
   times(A1,X,AX1),  times(B1,Y,BY1),  plus(AX1,BY1,C1),
   times(A2,X,AX2),  times(B2,Y,BY2),  plus(AX2,BY2,C2).
```

❖ Παράδειγμα:
- **X + 2*Y = 14**
- **X + Y = 9**

```
?- system(    equation(s(0),s(s(0)),s(s(...(s(0))))))))))),
              equation(s(0),s(0),s(s(...(s(0))))))))),
              Solution).

X = s(s(s(s(0))))
Y = s(s(s(s(s(0)))))
```

11 (+3 = 14)

8 (+3 = 9)

4

5