Εργαστήριο Τεχνητή Νοημοσύνη ΙΙ

Παύλος Πέππας

Τμήμα Ηλεκτρολόγων Μηχανικών

και Τεχνολογίας Υπολογιστών

Λύση Γρίφου του Einstein σε ASP

```
house(1..5).   col(r;g;b;w;y).  man(b;s;u;n;j).
drink(c;w;m;j;t).  cigatates(c;k;p;o;l).
pet(z;d;s;f;h).

1 { lives(X,Y): house(Y) } 1 :- man(X).
1 { smokes(X,Y): cigatates(Y) } 1 :- man(X).
1 { drinks(X,Y): drink(Y) } 1 :- man(X).
1 { hasPet(X,Y): pet(Y) } 1 :- man(X).
1 { hasColor(X,Y): col(Y) } 1 :- house(X).

:- lives(X,Y), lives(Z,Y), X != Z.
:- smokes(X,Y), smokes(Z,Y), X != Z.
:- drinks(X,Y), drinks(Z,Y), X != Z.
:- hasPet(X,Y), hasPet(Z,Y), X != Z.
:- hasColor(X,Y), hasColor(Z,Y), X != Z.

:- lives(b,Y), hasColor(Y,Z), Z != r.
:- hasPet(s,Y), Y != d.
:- lives(X,Y), hasColor(Y,g), drinks(X,Z), Z != c.
:- drinks(u,Z), Z != t.
```

```
:- hasColor(X,w), hasColor(Y,g), Y != X+1.
:- smokes(X,o), hasPet(X,Y), Y != s.
:- lives(X,Y), hasColor(Y,y), smokes(X,Z), Z != k.
:- lives(X,3), drinks(X,Y), Y != m.
:- lives(n,Y), Y != 1.
:- lives(X1,Y1), lives(X2,Y2), hasPet(X1,f),
   smokes(X2,c), Y1 != Y2+1, Y2 != Y1+1.
:- lives(X1,Y1), lives(X2,Y2), hasPet(X1,h),
  smokes(X2,k), Y1 != Y2+1, Y2 != Y1+1.
:- smokes(X,l), drinks(X,Y), Y != j.
:- smokes(j,Y), Y != p.
:- lives(n,Y), hasColor(Z,b), Y != Z+1, Z != Y+1.


#show hasPet/2.
#show drinks/2.
```

Βασικό Συντακτικό ASP

Οι **σταθερές** ξεκινούν με πεζά γράμματα. Π.χ red, block35, hourse2

Οι **μεταβλητές** ξεκινούν με κεφαλαία γράμματα. Π.χ., X, Y1, Book.

Facts:

mother(vicky,jim).

Rules:

uncle(X,Y) :- male(X), siblings(X,Z), parent(Z,Y).

Βασικό Συντακτικό ASP

Οι **σταθερές** ξεκινούν με πεζά γράμματα. Π.χ red, block35, hourse2

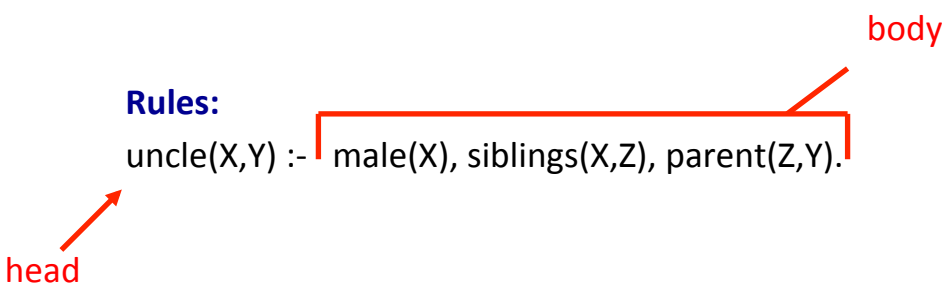
Οι **μεταβλητές** ξεκινούν με κεφαλαία γράμματα. Π.χ., X, Y1, Book.

Facts:

mother(vicky,jim).

Rules:

uncle(X,Y) :- male(X), siblings(X,Z), parent(Z,Y).



Βασικό Συντακτικό ASP

Οι **σταθερές** ξεκινούν με πεζά γράμματα. Π.χ red, block35, hourse2

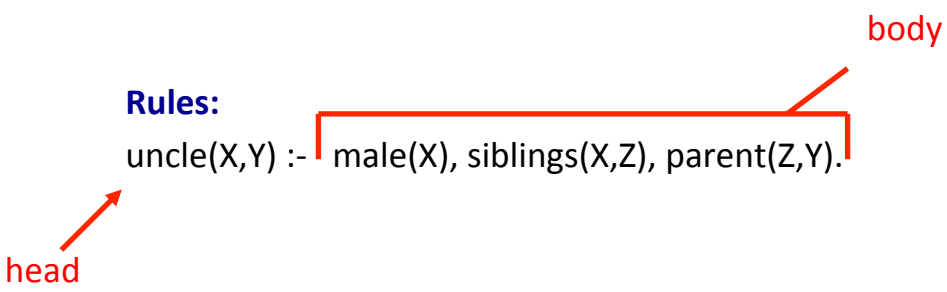
Οι **μεταβλητές** ξεκινούν με κεφαλαία γράμματα. Π.χ., X, Y1, Book.

Facts:

mother(vicky,jim).

Rules:

uncle(X,Y) :- male(X), siblings(X,Z), parent(Z,Y).



Integrity Constraints:

:- siblings(X,Y), X==Y.

Βασικό Συντακτικό ASP

Οι **σταθερές** ξεκινούν με πεζά γράμματα. Π.χ red, block35, hourse2

Οι **μεταβλητές** ξεκινούν με κεφαλαία γράμματα. Π.χ., X, Y1, Book.

Facts:

mother(vicky,jim).

Rules:

uncle(X,Y) :- male(X), siblings(X,Z), parent(Z,Y).

hasSibling(X) :- siblings(X,_).

Βασικό Συντακτικό ASP

Οι **σταθερές** ξεκινούν με πεζά γράμματα. Π.χ red, block35, hourse2

Οι **μεταβλητές** ξεκινούν με κεφαλαία γράμματα. Π.χ., X, Y1, Book.

Facts:

mother(vicky,jim).

Rules:

uncle(X,Y) :- male(X), siblings(X,Z), parent(Z,Y).

hasSibling(X) :- siblings(X,_).

commonFriend(X,Y) :- friends(X,_), friends(_,Y).

Βασικό Συντακτικό ASP

Οι **σταθερές** ξεκινούν με πεζά γράμματα. Π.χ red, block35, hourse2

Οι **μεταβλητές** ξεκινούν με κεφαλαία γράμματα. Π.χ., X, Y1, Book.

Facts:

mother(vicky,jim).

Rules:

uncle(X,Y) :- male(X), siblings(X,Z), parent(Z,Y).

hasSibling(X) :- siblings(X,_).

commonFriend(X,Y) :- friends(X,_), friends(Y,_).

Βασικό Συντακτικό ASP

Οι **σταθερές** ξεκινούν με πεζά γράμματα. Π.χ red, block35, hourse2

Οι **μεταβλητές** ξεκινούν με κεφαλαία γράμματα. Π.χ., X, Y1, Book.

Facts:

mother(vicky,jim).

Rules:

uncle(X,Y) :- male(X), siblings(X,Z), parent(Z,Y).

hasSibling(X) :- siblings(X,_).

commonFriend(X,Y) :- friends(X,_), friends(Y,_).

Integrity Constraints:

:- siblings(X,Y), X=Y.

Σταθερά Μοντέλα (Θετικού) Λογικού Προγράμματος

Στα πλαίσια του λογικού προγραμματισμού θα ονομάζουμε ***ερμηνεία*** Μ ένα σύνολο από ***ground atoms***. Μια ερμηνεία Μ είναι ***μοντέλο*** για ένα θετικό λογικό πρόγραμμα Π όταν ικανοποιεί όλους του κανόνες του. Το ***ελάχιστο*** μοντέλο ενός θετικού λογικού προγράμματος Π ονομάζεται ***σταθερό μοντέλο*** του Π

Παράδειγμα

```
mother(vicky,jim).

parent(X,Y) :-  mother(X,Y).
```

Σταθερά Μοντέλα (Θετικού) Λογικού Προγράμματος

Στα πλαίσια του λογικού προγραμματισμού θα ονομάζουμε ***ερμηνεία*** Μ ένα σύνολο από ***ground atoms***. Μια ερμηνεία Μ είναι ***μοντέλο*** για ένα θετικό λογικό πρόγραμμα Π όταν ικανοποιεί όλους του κανόνες του. Το ***ελάχιστο*** μοντέλο ενός θετικού λογικού προγράμματος Π ονομάζεται ***σταθερό μοντέλο*** του Π

Παράδειγμα

```
mother(vicky,jim).

parent(X,Y) :-  mother(X,Y).
```

Answer Set

```
mother(vicky,jim)
```

Σταθερά Μοντέλα (Θετικού) Λογικού Προγράμματος

Στα πλαίσια του λογικού προγραμματισμού θα ονομάζουμε ***ερμηνεία*** Μ ένα σύνολο από ***ground atoms***. Μια ερμηνεία Μ είναι ***μοντέλο*** για ένα θετικό λογικό πρόγραμμα Π όταν ικανοποιεί όλους του κανόνες του. Το ***ελάχιστο*** μοντέλο ενός θετικού λογικού προγράμματος Π ονομάζεται ***σταθερό μοντέλο*** του Π

Παράδειγμα

```
mother(vicky,jim).

parent(X,Y) :-  mother(X,Y).
```

Answer Set

```
mother(vicky,jim)
parent(vicky,jim)
```

Σταθερά Μοντέλα (Θετικού) Λογικού Προγράμματος

Στα πλαίσια του λογικού προγραμματισμού θα ονομάζουμε ***ερμηνεία*** Μ ένα σύνολο από ***ground atoms***. Μια ερμηνεία Μ είναι ***μοντέλο*** για ένα θετικό λογικό πρόγραμμα Π όταν ικανοποιεί όλους του κανόνες του. Το ***ελάχιστο*** μοντέλο ενός θετικού λογικού προγράμματος Π ονομάζεται ***σταθερό μοντέλο*** του Π

Παράδειγμα

```
mother(vicky,jim).

parent(X,Y) :-  mother(X,Y).
```

Answer Set

```
mother(vicky,jim)
parent(vicky,jim)
```

Παράδειγμα

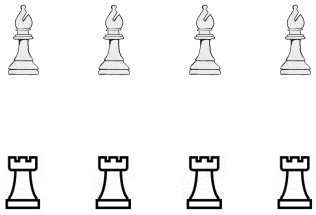
```
a :- b.
b :- a.
```

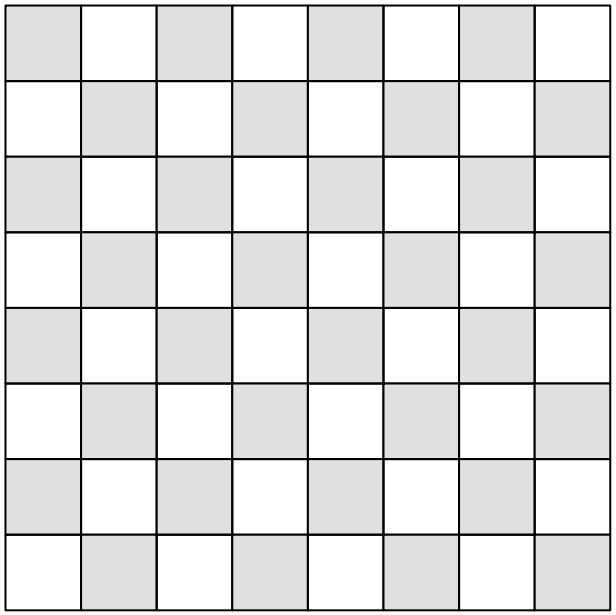
Answer Set

```

```

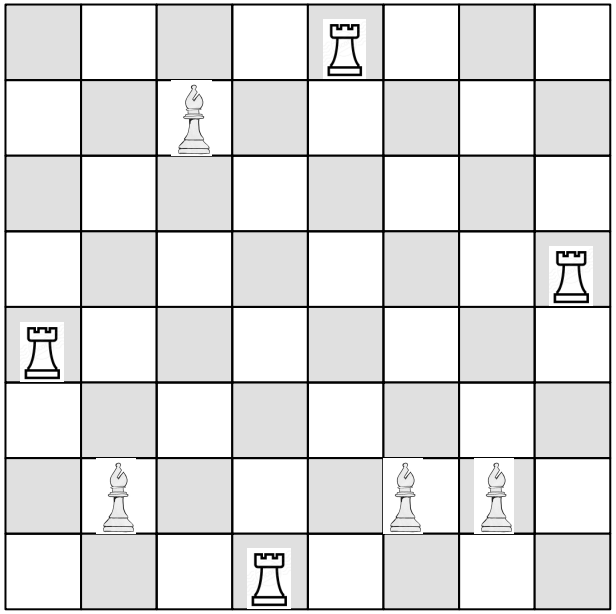
Άσκηση





Τοποθετήσετε τους πύργους και τους αξιωματικούς στην σκακιέρα χωρίς να αλληλο-απειλούνται.
Επιπλέον όλες οι στήλες θα πρέπει να έχουν ένα πύργο ή αξιωματικό.

Άσκηση



Τοποθετήσετε τους πύργους και τους αξιωματικούς στην σκακιέρα χωρίς να αλληλο-απειλούνται. Επιπλέον όλες οι στήλες θα πρέπει να έχουν ένα πύργο ή αξιωματικό.

Λύση

```
1 { tower(I,X,Y): X=1..8, Y=1..8 } 1 :- I=1..4.
1 { bishop(I,X,Y): X=1..8, Y=1..8 } 1 :- I=1..4.

occupied(Y) :- tower(_,_,Y), Y=1..8.
occupied(Y) :- bishop(_,_,Y), Y=1..8.

:- tower(I,X,Y), tower(J,X,Z),  I != J.
:- tower(I,X,Y), tower(J,Z,Y),  I != J.
:- bishop(I,X,Y), bishop(J,Z,W), X-Z == Y-W,  I != J.
:- bishop(I,X,Y), bishop(J,Z,W), Z-X == Y-W,  I != J.
:- tower(I,X,Y), bishop(J,X,W).
:- tower(I,X,Y), bishop(J,Z,Y).
:- tower(I,X,Y), bishop(J,Z,W), Z-X == W-Y.
:- tower(I,X,Y), bishop(J,Z,W), Z-X == Y-W.
:- tower(I,X,Y), bishop(J,Z,W), X-Z == W-Y.
:- tower(I,X,Y), bishop(J,Z,W), X-Z == Y-W.
:- not occupied(Y), Y=1..8.

#show tower/3.
#show bishop/3.
```