Εργαστήριο Τεχνητή Νοημοσύνη ΙΙ

Παύλος Πέππας

Τμήμα Ηλεκτρολόγων Μηχανικών

και Τεχνολογίας Υπολογιστών

Ισχυρή Άρνηση σε ASP

Μπορούμε να υλοποιήσουμε την κλασική (ισχυρή) άρνηση σε ASP (μόνο) για ατομικούς τύπους.

Π.χ.

```
a.
c  :-  a, -b.
```

Ισχυρή Άρνηση σε ASP

Μπορούμε να υλοποιήσουμε την κλασική (ισχυρή) άρνηση σε ASP (μόνο) για ατομικούς τύπους.

Π.χ.

```
a.
c :- a, -b.
```

είναι ουσιαστικά ισοδύναμο με το

```
a.
c :- a, -b.

a  :- a, -a.      a  :- b, -b.      a  :- c, -c.
-a :- a, -a.      -a :- b, -b.      -a :- c, -c.
b  :- a, -a.      b  :- b, -b.      b  :- c, -c.
-b :- a, -a.      -b :- b, -b.      -b :- c, -c.
c  :- a, -a.      c  :- b, -b.      c  :- c, -c.
-c :- a, -a.      -c :- b, -b.      -c :- c, -c.
```

Διαφορά των Δύο Ειδών Άρνηση

$\Pi_1 = \left\{ \begin{array}{l} \text{bird(tweety).} \\ \text{flies(X) :- bird(X), -penguin(X).} \end{array} \right\}$

$\Pi_2 = \left\{ \begin{array}{l} \text{bird(tweety).} \\ \text{flies(X) :- bird(X), not penguin(X).} \end{array} \right\}$

Διαφορά των Δύο Ειδών Άρνηση

$\Pi_1 = \left\{ \begin{array}{l} \text{bird(tweety).} \\ \text{flies(X) :- bird(X), -penguin(X).} \end{array} \right\}$

$\Pi_2 = \left\{ \begin{array}{l} \text{bird(tweety).} \\ \text{flies(X) :- bird(X), not penguin(X).} \end{array} \right\}$

M = { bird(tweety), flies(tweety) }

Διαφορά των Δύο Ειδών Άρνηση

$$\Pi_1 = \left\{ \begin{array}{l} \text{bird(tweety).} \\ \text{flies(X) :- bird(X), -penguin(X).} \end{array} \right\}$$

$$\Pi_2 = \left\{ \begin{array}{l} \text{bird(tweety).} \\ \text{flies(X) :- bird(X), not penguin(X).} \end{array} \right\}$$

M = { bird(tweety), flies(tweety) }

$\mathbf{M} \not\models_{\mathbf{ASP}} \mathbf{\Pi_1}$

Διαφορά των Δύο Ειδών Άρνηση

$$\Pi_1 = \left\{ \begin{array}{l} \text{bird(tweety).} \\ \text{flies(X) :- bird(X), -penguin(X).} \end{array} \right\}$$

$$\Pi_2 = \left\{ \begin{array}{l} \text{bird(tweety).} \\ \text{flies(X) :- bird(X), not penguin(X).} \end{array} \right\}$$

M = { bird(tweety), flies(tweety) }

$\mathbf{M} \not\models_{\mathbf{ASP}} \mathbf{\Pi_1}$

$$\Pi_2^M = \left\{ \begin{array}{l} \text{bird(tweety).} \\ \text{flies(X) :- bird(X).} \end{array} \right\}$$

Διαφορά των Δύο Ειδών Άρνηση

$$\Pi_1 = \left\{ \begin{array}{l} \text{bird(tweety).} \\ \text{flies(X) :- bird(X), -penguin(X).} \end{array} \right\}$$

$$\Pi_2 = \left\{ \begin{array}{l} \text{bird(tweety).} \\ \text{flies(X) :- bird(X), not penguin(X).} \end{array} \right\}$$

M = { bird(tweety), flies(tweety) }

$\mathbf{M} \nvDash_{ASP} \mathbf{\Pi_1}$

$\mathbf{M} \vDash_{ASP} \mathbf{\Pi_2}$

$$\Pi_2^M = \left\{ \begin{array}{l} \text{bird(tweety).} \\ \text{flies(X) :- bird(X).} \end{array} \right\}$$

Μη-Μονοτονικός Συμπερασμός σε ASP

$$\Pi = \left\{ \begin{array}{l} \text{bird(zippy).} \\ \text{bird(tweety).} \\ \text{-flies(zippy).} \\ \text{flies(X) :- bird(X), not –flies(X).} \end{array} \right\}$$

Μη-Μονοτονικός Συμπερασμός σε ASP

M = { bird(zippy), -flies(zippy), bird(tweety), flies(tweety) }

$$\Pi = \left\{ \begin{array}{l} \text{bird(zippy).} \\ \text{bird(tweety).} \\ \text{-flies(zippy).} \\ \text{flies(X) :- bird(X), not –flies(X).} \end{array} \right\}$$

$$\Pi^M = \left\{ \begin{array}{l} \text{bird(zippy).} \\ \text{bird(tweety).} \\ \text{-flies(zippy).} \\ \text{flies(tweety) :- bird(tweety).} \end{array} \right\}$$

Μη-Μονοτονικός Συμπερασμός σε ASP

M = { bird(zippy), -flies(zippy), bird(tweety), flies(tweety) }

Π =
```
bird(zippy).
bird(tweety).
-flies(zippy).
flies(X)  :-  bird(X), not –flies(X).
```

$Π^M$ =
```
bird(zippy).
bird(tweety).
-flies(zippy).
flies(tweety)  :-  bird(tweety).
```

Π =
```
q(a).
r(a).
p(X) :- q(X), not -p(X).
-p(X)  :-  r(X), not p(X).
```

Μη-Μονοτονικός Συμπερασμός σε ASP

Π =

```
bird(zippy).
bird(tweety).
-flies(zippy).
flies(X)  :-  bird(X), not –flies(X).
```

M = { bird(zippy), -flies(zippy), bird(tweety), flies(tweety) }

Π^M =

```
bird(zippy).
bird(tweety).
-flies(zippy).
flies(tweety)  :-  bird(tweety).
```

Π =

```
q(a).
r(a).
p(X) :- q(X), not -p(X).
-p(X)  :-  r(X), not p(X).
```

M = { p(a), q(a), r(a) }

Π^M =

```
q(a).
r(a).
p(a) :- q(a).
```

M = { -p(a), q(a), r(a) }

Π^M =

```
q(a).
r(a).
-p(a)  :-  r(a).
```

Μη-Μονοτονικός Συμπερασμός σε ASP

M = { bird(zippy), -flies(zippy), bird(tweety), flies(tweety) }

Π = { bird(zippy). bird(tweety). -flies(zippy). flies(X) :- bird(X), not –flies(X). }

$Π^M$ = { bird(zippy). bird(tweety). -flies(zippy). flies(tweety) :- bird(tweety). }

Π = { q(a). r(a). p(X) :- q(X), not -p(X). -p(X) :- r(X), not p(X). }

M = { p(a), q(a), r(a) }

$Π^M$ = { q(a). r(a). p(a) :- q(a). }

M = { -p(a), q(a), r(a) }

$Π^M$ = { q(a). r(a). -p(a) :- r(a). }

- Εύπιστη (credulous) Προσέγγιση: Επιλέγουμε τυχαία ένα answer set.
- Δύσπιστη (skeptical) Προσέγγιση: Πιστεύουμε μόνο ότι ισχύει σε όλα τα answer sets.

Μη-Μονοτονικός Συμπερασμός σε ASP
Περισσότερα Παραδείγματα

Π =
```
penguin(zippy).
bird(X) :- penguin(X).
flies(X)  :-  bird(X), not –flies(X).
-flies(X) :- penguin(X), not flies(X).
```

Μη-Μονοτονικός Συμπερασμός σε ASP

Περισσότερα Παραδείγματα

Π =

```
penguin(zippy).
bird(X) :- penguin(X).
flies(X)  :-  bird(X), not –flies(X).
-flies(X) :- penguin(X), not flies(X).
```

Answer Sets:

M1 = { bird(zippy), penguin(zippy), flies(zippy) } ✗

M2 = { bird(zippy), penguin(zippy), -flies(zippy) }

Μη-Μονοτονικός Συμπερασμός σε ASP
Περισσότερα Παραδείγματα

Π =
```
penguin(zippy).
bird(X) :- penguin(X).
flies(X)  :-  bird(X), not –flies(X).
-flies(X) :- penguin(X), not flies(X).
```

Answer Sets:

M1 = { bird(zippy), penguin(zippy), flies(zippy) } ✗

M2 = { bird(zippy), penguin(zippy), -flies(zippy) }

Π =
```
penguin(zippy).
bird(X) :- penguin(X).
flies(X)  :-  bird(X), –penguin(X), not –flies(X).
-flies(X) :- penguin(X), not flies(X).
```

Answer Sets:

M2 = { bird(zippy), penguin(zippy), -flies(zippy) }

Μη-Μονοτονικός Συμπερασμός σε ASP
Περισσότερα Παραδείγματα

Π =
```
penguin(zippy).
bird(X) :- penguin(X).
flies(X)  :-  bird(X), not –flies(X).
-flies(X) :- penguin(X), not flies(X).
```

Answer Sets:

M1 = { bird(zippy), penguin(zippy), flies(zippy) } ✗

M2 = { bird(zippy), penguin(zippy), -flies(zippy) }

Π =
```
penguin(zippy).
bird(X) :- penguin(X).
flies(X)  :-  bird(X), –penguin(X), not –flies(X).
-flies(X) :- penguin(X), not flies(X).
bird(tweety).
```

Answer Sets:

M = { bird(zippy), bird(tweety),
penguin(zippy), -flies(zippy) } ✗

Μη-Μονοτονικός Συμπερασμός σε ASP
Περισσότερα Παραδείγματα

Π =

```
penguin(zippy).
bird(X) :-penguin(X).
flies(X)  :-  bird(X), not –flies(X).
-flies(X) :- penguin(X), not flies(X).
```

Answer Sets:

M1 = { bird(zippy), penguin(zippy), flies(zippy) }

M2 = { bird(zippy), penguin(zippy), -flies(zippy) }

Π =

```
penguin(zippy).
bird(X) :- penguin(X).
flies(X)  :-  bird(X), not abBird(X).
-flies(X) :- penguin(X), not abPeng(X).
bird(tweety).

abPeng(X) :- penguin(X), flies(X).
abBird(X) :- penguin(X), not abPeng(X).
```

Answer Sets:

M = { bird(zippy), penguin(zippy),
bird(Tweety),
abBird(zippy),
-flies(zippy), flies(Tweety) }

Μη-Μονοτονικός Συμπερασμός σε ASP
Περισσότερα Παραδείγματα

Π =

```
penguin(zippy).
bird(X) :-penguin(X).
flies(X)  :-  bird(X), not –flies(X).
-flies(X) :- penguin(X), not flies(X).
```

Answer Sets:

M1 = { bird(zippy), penguin(zippy), flies(zippy) }

M2 = { bird(zippy), penguin(zippy), -flies(zippy) }

Π =

```
penguin(zippy).
bird(X) :- penguin(X).
flies(X)  :-  bird(X), not abBird(X).
-flies(X) :- penguin(X), not abPeng(X).
bird(tweety).
baby(tweety).

abBird(X) :- bird(X), baby(X).
abPeng(X) :- penguin(X), flies(X).
abBird(X) :- penguin(X), not abPeng(X).
```

Answer Sets:

M = { bird(zippy), penguin(zippy),

bird(Tweety), baby(tweety),

abBird(zippy), abBird(tweety),

-flies(zippy), ~~flies(Tweety)~~ }

Μη-Μονοτονικός Συμπερασμός σε ASP

Περισσότερα Παραδείγματα

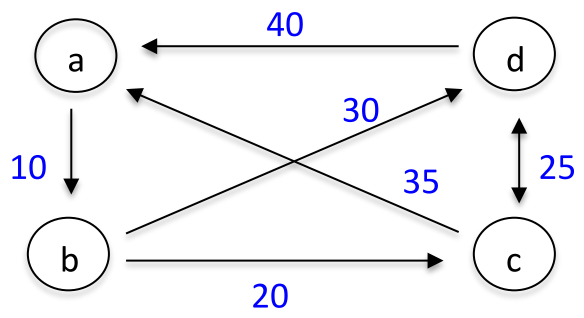
Π =

```
penguin(zippy).
bird(X) :- penguin(X).
flies(X)  :-  bird(X), not abBird(X).
-flies(X) :- penguin(X), not abPeng(X).
bird(tweety).
baby(tweety).
veryFit(zippy).

abPeng(X) :- penguin(X), veryFit(X).
abBird(X) :- bird(X), baby(X).
abPeng(X) :- penguin(X), flies(X).
abBird(X) :- penguin(X), not abPeng(X).
```

Answer Sets:

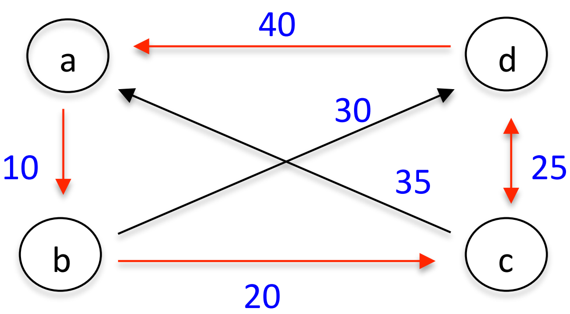
M = { bird(zippy), penguin(zippy), veryFit(zippy),
bird(Tweety), baby(tweety),
~~abBird(zippy)~~, abBird(tweety), abPeng(zippy),
~~-flies(zippy)~~, ~~flies(Tweety)~~, flies(zippy) }

Travelling Salesman



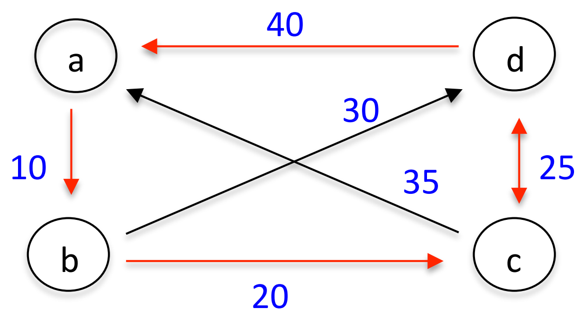
Γράψτε πρόγραμμα ASP που θα επιλύει το πρόβλημα του πλανόδιου πωλητή.

Λύση



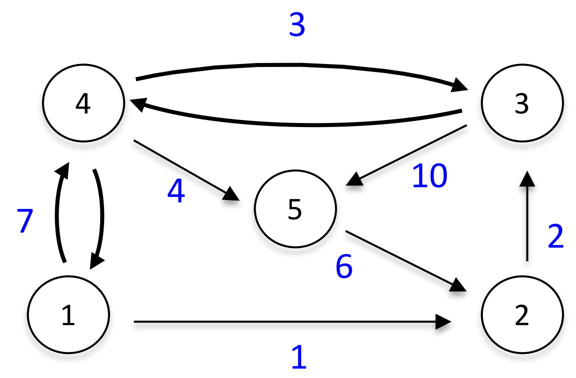

```
#const n=4.
city(a;b;c;d).
road(a,b,10). road(b,c,20). road(c,d,25). road(d,c,25). road(d,a,40). road(b,d,30). road(c,a,35).

% Επιλογή Κινήσεων
1 { at(C,T): city(C) } 1 :- T=1..n.

% Στοχος
visited(C) :- at(C,T), T=1..n.
:- city(C), not visited(C).

:- at(C1,T), at(C2,T+1), not road(C1,C2,_).
:- at(C1,n), at(C2,1), not road(C1,C2,_).

#minimize { D,C1,C2 : at(C1,T), at(C2,T+1), road(C1,C2,D); D,C1,C2 : at(C1,n), at(C2,1), road(C1,C2,D) }.
#show at/2.
```

Εκτέλεση




```
[<pavlos>$ clingo salesman.lp
clingo version 5.5.0
Reading from salesman.lp
Solving...
Answer: 1
at(a,4) at(b,1) at(c,3) at(d,2)
Optimization: 100
Answer: 2
at(a,4) at(b,1) at(c,2) at(d,3)
Optimization: 95
OPTIMUM FOUND

Models       : 2
  Optimum    : yes
Optimization : 95
Calls        : 1
Time         : 0.004s (Solving: 0.00s 1st Model: 0.00s Unsat: 0.00s)
CPU Time     : 0.004s
```

Εύρεση Βέλτιστης Διαδρομής



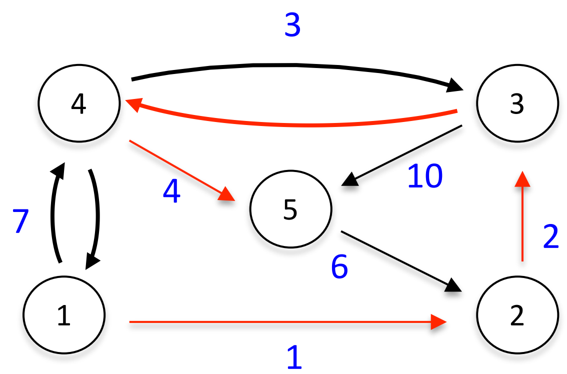
Γράψτε πρόγραμμα ASP που θα βρίσκει την βέλτιστη διαδρομή από την κορυφή 1 στην κορυφή 5.

Χρησιμοποιήσετε τα κατηγορήματα:

- node(X) για να δηλώσετε πως το X είναι κορυφή το γράφου,
- edge(X,Y,W) για να δηλώνει πως ο γράφος έχει ακμή που συνδέει την κορυφή X με την Y με κόστος W,

καθώς και όποια άλλα κατηγορήματα θεωρείται αναγκαία.

Λύση Α

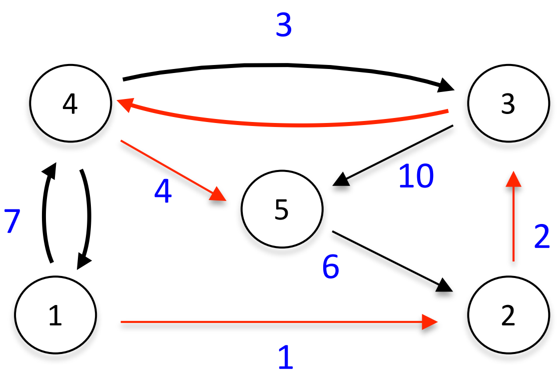


Το at(X,T) δηλώνει πως η βέλτιστη διαδρομή περνάει από την κορυφή X την στιγμή T.

```
#const n=5.
#const start=1.
#const end=5.

node(1..n).

edge(1,2,1). edge(1,4,7).
edge(2,3,2).
edge(3,4,3). edge(3,5,10).
edge(4,1,7). edge(4,3,3). edge(4,5,4).
edge(5,2,6).
```

```
0 { at(X,T): node(X) } 1 :- T=2..n.

at(start,1).
:- not at(end,_).

:- at(X,T), at(Y,T+1), not edge(X,Y,_).
:- not at(_,T), at(_,T+1), T=1..n-1.

#minimize { D,X,Y: at(X,T), at(Y,T+1), edge(X,Y,D) }.
#show at/2.
```

Λύση Β

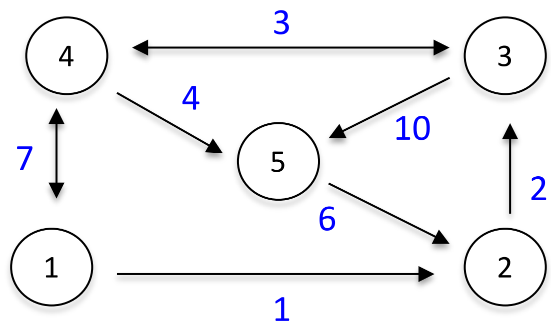


- Το inPath(X,Y) δηλώνει πως η ακμή (X,Y) βρίσκεται στο βέλτιστο μονοπάτι.
- Το reachable(X,Y) δηλώνει πως μέσω της βέλτιστης διαδρομής μπορούμε από την κορυφή X να φτάσουμε στην Y.

```
#const start=1.
#const end=5.

edge(1,2,1).   edge(1,4,7).
edge(2,3,2).
edge(3,4,3).   edge(3,5,10).
edge(4,1,7).   edge(4,3,3).   edge(4,5,4).
edge(5,2,6).
```

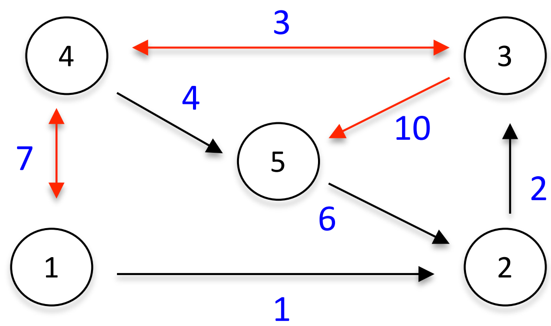
```
{ inPath(X,Y) : edge(X,Y,_) }.

reachable(X,Y) :- inPath(X,Y).
reachable(X,Y) :- reachable(X,Z), inPath(Z,Y).

:- not reachable(start,end).

#minimize{Z,X,Y: inPath(X,Y), edge(X,Y,Z)}.
#show inPath/2.
```

Εύρεση Χειρότερου Μονοπατιού



Γράψτε πρόγραμμα ASP που θα βρίσκει την χειρότερη ακυκλική διαδρομή από την 1 στην 5.

Λύση




```
#const n=5.
#const start=1.
#const end=5.

node(1..n).

edge(1,2,1). edge(1,4,7).
edge(2,3,2).
edge(3,4,3). edge(3,5,10).
edge(4,1,7). edge(4,3,3). edge(4,5,4).
edge(5,2,6).
```

```
0 { at(X,T): node(X) } 1 :- T=2..n.

at(start,1).
:- not at(end,_).

:- at(X,T), at(Y,T+1), not edge(X,Y,_).
:- at(X,T1), at(X,T2), T1 != T2.
:- at(end,T1), at(_,T2), T1<T2.

#maximize { D,X,Y: at(X,T), at(Y,T+1), edge(X,Y,D) }.
#show at/2.
```