

a-1) Si consideri il seguente programma logico e se ne calcolino gli answer set, illustrando adeguatamente il procedimento seguito.

```

a(Y) :- c(Y,X), not v(Y). 2
c(X,Z) :- a(X), v(Y), Z=X+Y. 3
v(Y) v notv(Y) :- not a(Y), c(Y,Z). 1

c(1,2).
c(4,5).

```

a-2) Si aggiunga il seguente strong constraint al programma del punto precedente.

```

:- c(4,Y), D=#sum{Z:v(Z)}, E=#sum{H:a(H)}, Y=D+E.

```

Come influisce sulle soluzioni del programma? Perché? Motivare adeguatamente la risposta.

$\alpha_1$

$$\begin{aligned} & \checkmark(1) \mid \text{not } v(1) : - \text{not } a(1), c(1,2). \\ & \checkmark(4) \mid \text{not } v(4) : - \text{not } a(4), c(4,5). \\ & a(1) : - \cancel{c(1,2)}, \text{not } v(1). \\ & a(4) : - \cancel{c(4,5)}, \text{not } v(4). \\ & \cancel{c(1,2)} : - a(1), v(1), 2=1+1. \quad \% a(1) \text{ e } v(1) \text{ non possono essere} \\ & \cancel{c(1,5)} : - a(1), v(4), 5=1+4. \quad \% \text{veri contemporaneamente} + c(1,2) \text{ è già EDB} \\ & \cancel{c(4,5)} : - a(4), v(1), 5=4+1. \quad \% c(4,5) \text{ è EDB} \\ & \cancel{c(4,8)} : - a(4), v(4), 8=4+4. \quad \% a(4) \text{ e } v(4) \text{ non possono} \\ & \quad \text{essere veri contemporaneamente} \end{aligned}$$

$$\cancel{v(1)} \mid \text{not } v(1) : - \text{not } a(1), c(1,5).$$

AS: {

$$\begin{aligned} & A1 \{ \dots \text{EDB} \dots, v(1), v(4) \} \\ & A2 \{ \dots \text{EDB} \dots, v(1), a(4) \} \\ & A3 \{ \dots \text{EDB} \dots, v(4), a(1), c(1,5) \} \\ & A4 \{ \dots \text{EDB} \dots, a(1), a(4) \} \end{aligned}$$

$\alpha_2$

$$:- c(4,Y), D=\#\sum\{Z:v(Z)\}, E=\#\sum\{H:a(H)\}, Y=D+E.$$

inserendo questo strong constraint, vengono scartati TUTTI gli AS, in quanto la somma degli argomenti delle 'v' e delle 'a' è sempre uguale a 5.

b) Si consideri ora un programma P (non è necessario sapere come è fatto) i cui answer set sono già stati calcolati e sono riportati di seguito.

**A1:** {q(2,3), q(4,2), p(4), f(3), m(1), m(3), f(2), m(2), f(4)}

**A2:** {q(2,3), q(4,2), p(4), f(3), m(1), m(3), f(2), p(2)}

**A3:** {q(2,3), q(4,2), p(4), f(3), m(1), p(3)}

Si supponga di aggiungere i seguenti weak constraint al programma P. Si calcoli quale sarebbe il costo di ognuno degli answer set riportati sopra, e si indichi quello ottimo, commentando il procedimento seguito.

:~ m(X), not l(X).	[2@X,X]	% dlv syntax: [2:X]
:~ p(X), f(X).	[1@X,X]	% dlv syntax: [1:X]

A1: 2@1 2@2 2@3 1@4

:~ m(1), not l(1).	[2@1,1]
:~ m(2), not l(2).	[2@2,2]
:~ m(3), not l(3).	[2@3,3]
:~ p(4), f(4).	[1@4,4]

A2: 2@1 1@2 2@3

:~ m(1), not l(1).	[2@1,1]
:~ m(3), not l(3).	[2@3,3]
:~ p(2), f(2).	[1@2,2]

A3: 2@1 1@3 OPTIMUM

:~ m(1), not l(1).	[2@1,1]
:~ p(3), f(3).	[1@3,3]

L'AS ottimo è A3.

**Esercizio 2.** Sia dato un grafo orientato  $G = \langle V, E \rangle$  tale che gli archi in  $E$  siano pesati (pesi solo positivi), e siano presenti in  $V$  due nodi speciali detti sorgente e destinazione (si può assumere che da un nodo  $X$  ad un nodo  $Y$  ci sia al più un arco). Determinare un insieme di archi che devono essere rimossi da  $G$  affinché il grafo ottenuto dopo la rimozione rispetti tutte le seguenti condizioni:

- s1. La destinazione NON deve essere raggiungibile dalla sorgente.
- s2. Non devono essere presenti cicli.
- s3. Ogni nodo deve avere un numero di archi entranti non superiore a 3.

Inoltre, dovrebbero essere preferite le soluzioni che rispettano possibilmente i seguenti requisiti:

- w1. Per ogni nodo, dovrebbe essere rimosso al più un arco uscente.
- w2. Se per un certo nodo  $X$  più di un arco uscente deve essere rimosso, allora dovrebbero essere preferite le soluzioni in cui il peso totale degli archi uscenti rimossi per  $X$  è minimo.

Si noti che tra w1 e w2, è più importante soddisfare w1. Si noti inoltre che, la preferenza w2 deve essere tenuta in considerazione solo per i nodi per i quali non è possibile soddisfare w1.

Modello dei dati in INPUT:

- node( $X$ )  $\leftarrow$  i nodi in  $V$  del grafo in input.
- arc( $X, Y, W$ )  $\leftarrow$  gli archi in  $E$  del grafo in input, dove  $W$  indica il peso.
- source( $X$ )  $\leftarrow$  il nodo sorgente in  $V$ .
- destination( $X$ )  $\leftarrow$  il nodo destinazione in  $V$ .

- newArc( $X, Y, W$ ) | newArc( $X, Y, W$ ) :- arc( $X, Y, W$ ).  
:- #count{ $X, Y, W$ , newArc( $X, Y, W$ )} = 0.

% s1

reaches( $X, Y$ ) :- newArc( $X, Y, -$ ).  
reaches( $X, Z$ ) :- reaches( $X, Y$ ), newArc( $Y, Z$ ).  
:- source( $S$ ), destination( $D$ ), reaches( $S, D$ ).

% s2

:- reaches( $X, Y$ ), reaches( $Y, X$ ). reaches( $X, X$ ).

% s3

:- node( $X$ ), #count{ $X, Y, W$ : newArc( $Y, X, W$ )} > 3.

% w1

:- node( $X$ ), #count{ $X, Y, W$ : newArc( $X, Y, W$ )} > 3.

ViolatingW1( $X$ ).:- node( $X$ ),  $\neg$  woni( $X, i, w$ ) .  $\rightarrow$  newArc( $X, i, w$ ),  $i > 1$ .  
:~ violatingW1( $X$ ). [1@2,  $X$ ]

% w2

:~ violatingW1( $X$ ),  $\neg$  newArc( $X, Y, w$ ) [w@1,  $XY$ ]

