Prova d'esame del 13/02/2014

Esercizio 1. Svolgere tutti i punti.

a) Si consideri il seguente programma logico. Se ne calcolino gli answer set illustrando adeguatamente il procedimento seguito.

P.
```
vacanza(1). vacanza(2). libero(3,1). libero(2,2).

piove(X) v gita(X) :- vacanza(X).
vacanza(X) :- libero(X,Y), not piove(Y).

:- #count{X:gita(X)}<2, #count{Y:vacanza(Y)}>2.
```

b) Si aggiunga il seguente weak constraint:

```
:~ piove(X), vacanza(Y). [1:X]
```

Calcolare quindi gli answer set riportando per ciascuno il costo. Indicare quindi quello ottimo (o quelli ottimi, se più di uno).

c) Si aggiunga ancora il seguente weak constraint.

```
:~ not gita(X), not piove(X), libero(X,Y). [Y:X]
```

Qual è adesso l'answer set ottimo (o quelli ottimi, se più di uno)? Qual è il costo di tutti gli altri answer set?

Esercizio 2. Il nostro amico Ciccio Pasticcio si è appassionato al free climbing. Purtroppo, è uno sport non esente da rischi… e così, cadendo, è finito in ospedale. Niente di grave, per fortuna… anzi: la sua esperienza gli ha consentito di osservare il funzionamento di una complessa macchina come quella di un grosso ospedale. In particolare, lo ha incuriosito il modo in cui i pazienti vengono ricevuti e assistiti; la sua deformazione professionale lo ha quindi spinto ad immaginare in che modo il tutto possa essere automatizzato… grazie ad un programma DLV. Provate ad immaginare in che modo questo programma possa essere concepito, tenendo conto delle considerazioni riportate di seguito.

- Esiste un certo insieme, piuttosto ampio, di tipologie di "problemi".
- Quando un paziente arriva, viene immediatamente "registrato" con i propri dati ed i "problemi" che presenta.
- Ciascun problema ha un grado di "urgenza", indicato con un numero: più è alto il numero, più urgente è il problema.
- L'ospedale è strutturato in una serie di reparti, ciascuno dei quali:
  - è attrezzato a gestire un certo numero di tipologie di "problemi";
  - ha una certa capienza;
  - (per semplicità) può curare un solo paziente alla volta (quindi è necessario formare delle liste di attesa).
- Se un paziente finisce in un reparto, si suppone che quel reparto tratterà tutti i problemi di quel paziente per cui è accettato. Ad esempio, se un paziente presentasse 3 problemi: (i) mano slogata, (ii) escoriazioni e (iii) contusioni, e fosse assegnato ad un reparto che è in grado di gestire le 5 tipologie di problemi (a) escoriazioni, (b) contusioni, (c) fratture, (d) sanguinamento, (e) pressione bassa, allora il reparto "affronterebbe" sia le escoriazioni che le contusioni del paziente, il quale però dovrebbe poi essere assegnato ad un altro reparto per curare la mano slogata.
- L'obiettivo del programma è l'assegnamento di ogni paziente ad un numero di reparti sufficiente a "curare" tutti i suoi problemi.

Prova d'esame del 13/02/2014

- Non è possibile che un paziente rimanga con un problema non "curato".
- Non è possibile assegnare un paziente ad un reparto che non curi nessuno dei suoi "problemi".
- Non è possibile assegnare un paziente a più di 4 reparti.
- Si desidera, per ciascun reparto, MASSIMIZZARE il numero di pazienti con problemi urgenti. Naturalmente, non si può penalizzare una soluzione per la mancata assegnazione di un paziente con problema urgente ad un reparto che non è attrezzato per trattare quel problema.
- Ancora più importante, si desidera, per ciascun paziente, MINIMIZZARE il numero di reparti presso i quali è costretto a "passare".

Modello dei dati in input:

problema(Nome,Urgenza) ← le tipologie di "problemi"
paziente(CodiceFiscale,Problema) ← i problemi di ciascun paziente
reparto(Nome,Problema) ← i problemi per cui ciascun reparto è attrezzato

Si noti che, in input, ci si possono attendere più istanze del predicato "paziente" per un singolo paziente (una istanza per ciascuno dei problemi che presenta); analogamente sarà per il predicato reparto (per ciascun reparto, tante istanze quanti sono i problemi che è in grado di curare). Ad esempio, nel caso riportato sopra, avremmo:

paziente(xxxyyy, mano_slogata). paziente(xxxyyy, escoriazioni). paziente(xxxyyy, contusioni).

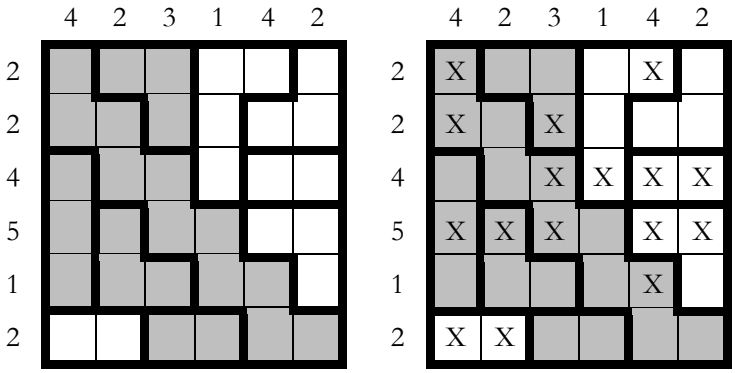
e poi

reparto(2c, escoriazioni). reparto(2c, contusioni). reparto(2c, fratture).
reparto(2c, sanguinamento. reparto(2c, pressione bassa).

Esercizio 3. Si scriva un programma DLV che consenta di risolvere istanze generiche del puzzle game denominato "Pari-Dispari", e descritto di seguito.

È data una griglia in cui ciascuna cella può essere bianca o grigia. La griglia è divisa in settori, ossia gruppi di celle contigue, tutte dello stesso colore. Nelle figure qui riportate i settori sono delimitati da bordi ingrossati.

Si devono inserire delle "X" in alcune celle della griglia, rispettando i vincoli indicati di seguito.



- Ogni settore grigio deve contenere un numero DISPARI di "X".
- Ogni settore bianco deve contenere un numero PARI di "X".
- Possono esserci settori "vuoti" (cioè tali che nessuna delle celle contenute contiene una "X", o, equivalentemente, settori con un numero ZERO di "X"); tuttavia, può esserci AL MASSIMO un settore "vuoto" per ciascun colore (bianco/grigio).
- Esternamente alla griglia sono riportati dei numeri interi, che indicano quante "X" devono comparire nella riga o colonna corrispondente.

ESEMPIO: Nelle figure sono riportate una istanza 6x6 con la corrispondente soluzione.