REPRESENTACIÓN DEL CONOCIMIENTO Y RAZONAMIENTO AUTOMÁTICO

Examen 25/5/2017. **Apellidos, nombre: . . . .**

1) Dado el siguiente programa lógico proposicional:

$$\begin{aligned} p &\leftarrow q, \mathit{not}\ r \\ q &\leftarrow p \\ r &\leftarrow \mathit{not}\ p \end{aligned}$$

Indica cuáles son sus modelos clásicos mediante una tabla de verdad. De entre los modelos clásicos, indica luego cuáles son modelos soportados (*supported models*) y, a su vez, cuáles de estos son modelos estables (*stable models*), justificando las respuestas.

Aunque podemos comprobar cada regla por separado, es fácil ver que la tercera regla es equivalente a $p \vee r$ mientras que la primera equivale a $(r \vee \neg q \vee p)$ y, por tanto, está subsumida por la tercera y se puede eliminar para obtener modelos clásicos.

p	q	r	$p \vee r$	$p \rightarrow q$	*modelo*
0	0	0	0	1	
0	0	1	1	1	×
0	1	0	0	1	
0	1	1	1	1	×
1	0	0	1	0	
1	0	1	1	0	
1	1	0	1	1	×
1	1	1	1	1	×

Es decir, obtenemos cuatro modelos clásicos $\{r\}$, $\{q, r\}$, $\{p, q\}$ y $\{p, q, r\}$. Para decidir si son *supported*, comprobamos si son puntos fijos del operador:
$T_P(I) = \{H \mid (H \leftarrow B) \in P, I \models B\}$
es decir, recopilar las cabezas de las reglas cuyo cuerpo es cierto en cada modelo I.

I	$T_P(I)$	*supported*
$\{r\}$	$\{r\}$	×
$\{q, r\}$	$\{r\}$	
$\{p, q\}$	$\{p, q\}$	×
$\{p, q, r\}$	$\{q\}$	

Por último, para ver si los soportados son también estables, calculamos el reducto. El reducto $P^{\{r\}}$ es el programa con las reglas $(q \leftarrow p)$ y $(r \leftarrow)$ que, tras aplicarlas exhaustivamente, dan como modelo mínimo $\{r\}$, por lo que $\{r\}$ **es modelo estable**. Por otro lado, el reducto $P^{\{p,q,r\}}$ es el programa con las reglas $(p \leftarrow q)$ y $(q \leftarrow p)$ y su modelo mínimo es $\emptyset$ por lo que no es modelo estable.

Otro modo de calcular los modelos soportados es usando la *completion*:

$$p \leftrightarrow q \wedge \neg r \qquad q \leftrightarrow p \qquad r \leftrightarrow \neg p$$

La segunda fórmula hace que p y q sean equivalentes y la tercera hace que el valor de p y q sea el opuesto a r. Con esto, la primera fórmula se vuelve redundante. Obtendríamos dos modelos: uno con p y q falsos y, por tanto, r cierto $\{r\}$; y otro con los dos ciertos $\{p, q\}$ y r falso.

2) Queremos pintar un mapa político con los colores rojo, verde y azul sin que se repitan colores entre países vecinos usando el predicado `pinta(Pais,Color)`. Para ello, nos proporcionan el siguiente programa ASP incompleto:

```
color(rojo;verde;azul).      pais(fr;de;be).
vecino(fr,de).               vecino(fr,be).      vecino(be,de).
vecino(X,Y) :- vecino(Y,X).
#show pinta/2.
```

Completa el programa para que genere las soluciones buscadas.

```
1 { pinta(X,C) : color(C) } 1 :- pais(X).
:- vecino(X,Y), pinta(X,C), pinta(Y,C).
```

¿Cuántas soluciones debe generar para el conjunto de países dado arriba? Como tenemos justo 3 países y todos son vecinos de todos, podemos colocarlos alfabéticamente y asignar un color distinto a cada uno. Núm. soluciones = permutaciones de 3 colores, es decir, $3! = 6$.

¿Cuántas reglas *ground* generará como máximo tu programa? Razona la respuesta.

El programa tiene 3 hechos para `color`, 3 hechos para `pais`, y 6 hechos para `vecino` (debido al cierre simétrico). La regla choice generará 3 casos ground (uno por país `X`) y la restricción generará 6×3, cada uno de los 6 hechos `vecino(X,Y)` por cada uno de los tres colores `C`. Así que el máximo de reglas sería $3{+}3{+}6{+}3{+}18 = 33$.

Para matrícula ...

Una respuesta más afinada: con la mitad de las constraints es suficiente. Cuando tenemos `vecino(fr,be)` y su simétrico `vecino(be,fr)` las constraints generadas son:
`:- pinta(fr,rojo), pinta(be,rojo)  :- pinta(be,rojo), pinta(fr,rojo)`
ya que los hechos sobre `vecino` ciertos ya no se incluyen en las reglas ground. Ambas reglas son equivalentes y el grounder podría eliminar una de ellas. Si lo hace, las reglas ground se reducirían a $3{+}3{+}6{+}3{+}9 = 24$.