REPRESENTACIÓN DEL CONOCIMIENTO Y RAZONAMIENTO AUTOMÁTICO
Examen 7/2017. **Apellidos, nombre:** .....

1) Dado el siguiente programa lógico proposicional:

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

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.

La primera regla corresponde a $r \wedge \neg q \rightarrow p$ que equivale a $\neg r \vee q \vee p$. La segunda sería $\neg p \rightarrow q$ que equivale a $p \vee q$ y que subsume a la primera regla (es más fuerte). Por tanto, para los modelos clásicos, podemos ignorar la primera regla. Por último, la tercera regla correspondería a $\neg p \rightarrow r$ que equivale a $p \vee r$ en lógica clásica.

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

Es decir, obtenemos cinco modelos clásicos $\{q, r\}$, $\{p\}$, $\{p, r\}$, $\{p, q\}$, 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*
$\{q, r\}$	$\{q, r\}$	×
$\{p\}$	$\emptyset$	
$\{p, r\}$	$\{p\}$	
$\{p, q\}$	$\emptyset$	
$\{p, q, r\}$	$\emptyset$	

Por último, para ver si el modelo soportado que hemos obtenido es también estable, calculamos el reducto. El reducto $P^{\{q,r\}}$ es el programa con las reglas $(q \leftarrow)$ y $(r \leftarrow)$ que, tras aplicarlas exhaustivamente, dan como modelo mínimo $\{q, r\}$, por lo que $\{q, r\}$ **es modelo estable**.

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

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

Las dos últimas fórmulas hacen que tanto r como q tengan el valor opuesto a p. Si tomamos p cierto, entonces r y q serían falsos y la primera fórmula no se podría satisfacer. Por tanto, la única posibilidad es tomar p falso y q y r ciertos, lo que satisface las tres fórmulas. Esto da como único modelo soportado $\{q, r\}$, que obtuvimos antes usando el operador T_P.

2) El Sudoku es una elaboración de un juego propuesto por Euler que consistía en rellenar un tablero cuadrado de $n \times n$ de modo que coloquemos en cada celda números de 1 a n sin repetirlos en ninguna fila ni en ninguna columna (el Sudoku fija $n = 9$ y añade además esa prohibición para bloques de 3×3). Para resolver el problema de Euler con $n = 3$ nos proponen, como punto de partida, el siguiente programa:

```
#const n=3.
digito(1..n). fila(1..n). columna(1..n).
#show celda/3.
1 {celda(X,Y,D): digito(D)} 1 :- fila(X),columna(Y).
```

¿Qué tipo de soluciones generaría este programa? ¿Cuántas generaría? Este programa asigna un número entre 1 y 3 a cada celda. Esto permite repetir números en la misma fila o columna. Como hay $3 \times 3 = 9$ celdas y, para cada una, podemos elegir entre 3 posibilidades, el número de soluciones sería 3^9.

Añade al programa de arriba las reglas que necesites para resolver el problema de Euler.

```
:- celda(X,Y,D), celda(X,Y1,D), Y<Y1.
:- celda(X,Y,D), celda(X1,Y,D), X<X1.
```

¿Cuántas reglas *ground* (incluyendo hechos) generará como máximo tu programa? Razona la respuesta.

El programa tiene 3 hechos para `fila`, 3 hechos para `columna`, y 3 hechos para `digito`. La regla choice generará 9 casos ground (uno por cada celda `X,Y`). La primera restricción, por cada fila `X` y por cada dígito `D` toma pares de números `Y<Y1` entre 1 y 3. Esto sólo genera tres casos: `Y=1`, `Y1=2`; `Y=1`, `Y1=3`; y `Y=2`, `Y1=3`. En total, 3 filas $\times$ 3 dígitos $\times$ 3 pares $=$ 27 reglas ground. La segunda restricción es totalmente simétrica y genera otros 27 casos. En total, tendremos 3+3+3+9+27+27=72 reglas ground.

Si, en lugar de haber usado `Y<Y1` hubiésemos escrito `Y!=Y1` entonces el número de pares `Y,Y1` pasaría a ser $3 \times 3 - 3 = 6$ (ya que los casos en son iguales son descartados). Eso haría que cada

restricción generase $3 \times 3 \times 6 = 54$ y el total de reglas se disparase a 3+3+3+9+54+54=126.