REPRESENTACIÓN DEL CONOCIMIENTO Y RAZONAMIENTO AUTOMÁTICO

Examen 17/7/2018. Apellidos, nombre:

Ejercicio 1. Dado el siguiente programa lógico proposicional *P*:

$$a \leftarrow not b$$
 (1)

$$b \leftarrow c, not b$$
 (2)

$$c \leftarrow not a$$
 (3)

1a) Indica cuáles son sus modelos clásicos mediante una tabla de verdad.

En lógica clásica proposicional, la primera regla (1) sería la fórmula $(\neg b \to a)$ que es equivalente a $(a \lor b)$. La segunda regla (2) sería, por tanto, $(\neg b \land c \to b)$ que es equivalente a $(b \lor \neg c)$. Por útlimo, es fácil ver que la tercera regla (3) es clásicamente equivalente a $(a \lor c)$. Los modelos clásicos son aquellos que cumplen la conjunción de las tres reglas:

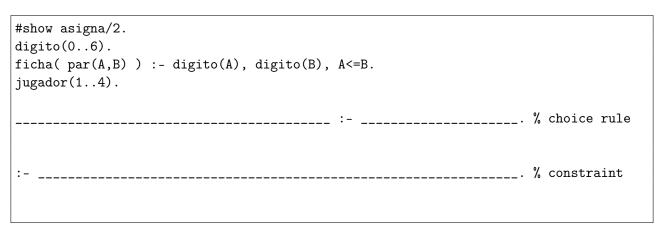
\boldsymbol{a}	b	c	$(1) \equiv a \vee b$	$(2) \equiv b \vee \neg c$	$(3) \equiv a \vee c$	$(1) \wedge (2) \wedge (3)$
0	0	0	0	1	0	0
0	0	1	0	0	1	0
0	1	0	1	1	0	0
0	1	1	1	1	1	1
1	0	0	1	1	1	1
1	0	1	1	0	1	0
1	1	0	1	1	1	1
1	1	1	1	1	1	1

Es decir, obtenemos cuatro modelos clásicos $I_0=\{b,c\},\,I_1=\{a\},\,I_2=\{a,b\}$ e $I_3=\{a,b,c\}$.

1b) Para cada modelo clásico I obtenido anteriormente, obtén el reducto P^{I} , el modelo mínimo de ese reducto e indica, a partir de él, si I es modelo estable ($stable\ model$).

modelo clásico I	reducto P^I	$ \ \ \ \text{modelo mínimo de } P^I $	$\begin{array}{c} \text{estable} \\ \text{(si/no)} \end{array}$
$I_0 = \{b, c\}$	$c \leftarrow$	$\{c\}$	No
$I_1 = \{a\}$	$\begin{array}{c} a \leftarrow \\ b \leftarrow c \end{array}$	$\{a\}$	Sí
$I_2 = \{a, b\}$		Ø	No
$I_3 = \{a, b, c\}$		Ø	No

Ejercicio 2. Se desea construir un programa ASP que genere los posibles repartos de 7 fichas a cada uno de los 4 jugadores para definir la posición inicial de una partida de dominó por parejas. Para ello, nos proporcionan el siguiente código incompleto:



donde la solución se mostrará en función del predicado asigna(J,F) siendo J un número de jugador y F una de las posibles fichas de la forma par(A,B) generadas por el predicado ficha arriba definido.

2a) ¿ Cuántos hechos se generan para el predicado ficha(F)? Se forman 28 fichas.

Aunque no se pide explicar el resultado, como aclaración adicional: en general, si tenemos n dígitos diferentes, el número de fichas viene dado por combinaciones con repetición de esos n dígitos tomados de dos en dos:

$$\binom{n+2-1}{2} = \frac{(n+2-1)!}{2! (n-1)!} = \frac{n (n+1)}{2}$$

y con n = 7 (dígitos del 0 al 6) obtenemos 56/2 = 28.

2b) Añade una choice rule para hacer que a cada jugador se le asignen exactamente 7 fichas diferentes

```
7 { asigna(J,F): ficha(F) } 7 :- jugador(J).
```

2c) Añade una constraint para que una misma ficha no sea asignada a dos jugadores distintos

```
:- asigna(J,F), asigna(K,F), J!=K.
```

2d) Una vez finalizado el programa, explica qué ocurriría si quitamos la condición A<=B de la regla para el predicado ficha de arriba. ¿Seguirían siendo correctas todas las soluciones obtenidas? No. La ficha de dominó se identifica por un par de dígitos, pero el orden entre ellos es irrelevante. La condición A<=B permite poner siempre en par(A,B) el dígito menor en primer lugar y evita así duplicar fichas. Si quitamos esa condición, tendríamos que, por ejemplo, par(2,3) y par(3,2) serían tratadas como fichas diferentes y un jugador podría recibir esas dos fichas entre las 7, lo que no tendría correspondencia con una configuración real del dominó (suponiendo que no hay fichas repetidas).