Solución declarativa del Sudoku mediante ASP

José A. Alonso Jiménez

Grupo de Lógica Computacional
Dpto. de Ciencias de la Computación e Inteligencia Artificial
Universidad de Sevilla
Sevilla, 17 de Junio de 2006 (versión de 28 de agosto de 2006)

Resumen

El objetivo del presente trabajo es presentar una solución declarativa del problema del Sudoku. Se presenta una especificación genérica en ASP ("Answer Set Programming") ([1]) y su utilización para resolver dos problemas concretos. Como sistema ASP se ha usado SMODELS ([3] y [4]). La especificación se basa en la presentada en [2].

Índice

1. Enunciado

El problema del Sudoku de orden $N \times N$ consiste en completar un tablero (subdividido en N cuadrados) de $N \times N$ casillas (dispuestas en N filas y N columnas) rellenando las celdas vacías con los números del 1 al N, de modo que no se repita ninguna cifra en cada fila, ni en cada columna, ni en cada cuadrado.

Ejemplo 1. En la figura 1 se muestra un Sudoku 4×4 y su solución.

1			
		2	
	3		
			4

1	2	4	3
3	4	**2**	1
4	**3**	1	2
2	1	3	**4**

Figura 1: Sudoku 4×4

Ejemplo 2. En la figura 2 se muestra un Sudoku 9×9 y su solución.

4		**5**			**2**			
	7			**6**			**1**	
				5			**7**	**9**
			5				**2**	**8**
8		**3**			**4**			
1			**9**			**3**		**5**
	2	**4**					**6**	
				7	**9**	**8**		
	9		**6**					**1**

4	1	**5**	7	9	**2**	6	8	3
3	**7**	9	4	**6**	8	5	**1**	2
2	8	6	3	**5**	1	4	**7**	**9**
9	4	7	**5**	3	6	1	**2**	**8**
8	5	**3**	1	2	**4**	7	9	6
1	6	2	**9**	8	7	**3**	4	**5**
5	**2**	**4**	8	1	3	9	**6**	7
6	3	1	2	**7**	**9**	**8**	5	4
7	**9**	8	**6**	4	5	3	3	**1**

Figura 2: Sudoku 9×9

2. Representación

Un Sudoku puede representarse mediante la relación `estado(X,Y,V)` que se verifica si el valor en la casilla de la fila `X` y la columna `Y` es `V`. Por ejemplo, el problema del Sudoku de la figura 1 se representa, en el fichero `pb_4x4_1.pro`, por

```
estado(1,1,1).
estado(2,3,2).
estado(3,2,3).
estado(4,4,4).
```

y el de la figura 2 se representa, en el fichero `pb_9x9_1.pro`, por

```
estado(1,1,4). estado(1,3,5). estado(1,6,2).
estado(2,2,7). estado(2,5,6). estado(2,8,1).
estado(3,5,5). estado(3,8,7). estado(3,9,9).
estado(4,4,5). estado(4,8,2). estado(4,9,8).
estado(5,1,8). estado(5,3,3). estado(5,6,4).
estado(6,1,1). estado(6,4,9). estado(6,7,3). estado(6,9,5).
estado(7,2,2). estado(7,3,4). estado(7,8,6).
estado(8,5,7). estado(8,6,9). estado(8,7,8).
estado(9,2,9). estado(9,4,6). estado(9,9,1).
```

3. Solución

En la siguiente sección se presenta el programa `sudoku_NxN.pro` que permite resolver con SMODELS el problema del Sudoku. Por ejemplo, la solución del Sudoku de la figura 1 se obtiene con

```
> lparse -W none -c n=4 sudoku_NxN.pro pb_4x4_1.pro | smodels 0
smodels version 2.28. Reading...done
Answer: 1
Stable Model:
  estado(1,1,1) estado(2,1,3) estado(3,1,4) estado(4,1,2)
  estado(1,2,2) estado(2,2,4) estado(3,2,3) estado(4,2,1)
  estado(2,3,2) estado(1,3,4) estado(3,3,1) estado(4,3,3)
  estado(2,4,1) estado(1,4,3) estado(3,4,2) estado(4,4,4)
False
Duration: 0.024
```

y la del Sudoku de la figura 2 se obtiene con

```
> lparse -W none sudoku_9x9_v2.pro pb_9x9_1.pro | smodels 0
smodels version 2.28. Reading...done
Answer: 1
Stable Model:
 estado(2,1,3) estado(1,1,4) estado(3,1,2) estado(4,1,9) estado(5,1,8)
 estado(6,1,1) estado(7,1,5) estado(8,1,6) estado(9,1,7) estado(1,2,1)
 estado(2,2,7) estado(3,2,8) estado(4,2,4) estado(5,2,5) estado(6,2,6)
 estado(7,2,2) estado(8,2,3) estado(9,2,9) estado(1,3,5) estado(2,3,9)
 estado(3,3,6) estado(4,3,7) estado(5,3,3) estado(6,3,2) estado(7,3,4)
 estado(8,3,1) estado(9,3,8) estado(2,4,4) estado(1,4,7) estado(3,4,3)
 estado(4,4,5) estado(5,4,1) estado(6,4,9) estado(7,4,8) estado(8,4,2)
 estado(9,4,6) estado(2,5,6) estado(1,5,9) estado(3,5,5) estado(4,5,3)
 estado(5,5,2) estado(6,5,8) estado(7,5,1) estado(8,5,7) estado(9,5,4)
 estado(1,6,2) estado(2,6,8) estado(3,6,1) estado(4,6,6) estado(5,6,4)
 estado(6,6,7) estado(7,6,3) estado(8,6,9) estado(9,6,5) estado(2,7,5)
 estado(1,7,6) estado(3,7,4) estado(4,7,1) estado(5,7,7) estado(6,7,3)
```

```
 estado(7,7,9) estado(8,7,8) estado(9,7,2) estado(2,8,1) estado(1,8,8)
 estado(3,8,7) estado(4,8,2) estado(5,8,9) estado(6,8,4) estado(7,8,6)
 estado(8,8,5) estado(9,8,3) estado(2,9,2) estado(1,9,3) estado(3,9,9)
 estado(4,9,8) estado(5,9,6) estado(6,9,5) estado(7,9,7) estado(8,9,4)
 estado(9,9,1)
False
Duration: 0.400
```

4. Programa

Definición 1. El orden del Sudoku es `NxN` (por defecto, `N` es 9).

```
const n=9.
```

Definición 2. `posicion(X)` se verifica si `X` es una posición de las filas o de las columnas; es decir, si `X` es un número entre 1 y `N`.

```
posicion(1..n).
```

Notación 1. Se usa `X`, `Y`, `Z`, `XA`, `XB`, `YA`, `YB` como variables sobre posiciones.

```
#domain posicion(X;Y;Z;XA;XB;YA;YB).
```

Definición 3. `valor(X)` se verifica si `X` es el valor de una casilla; es decir, si `X` es un número entre 1 y 9.

```
valor(1..n).
```

Notación 2. Se usa `V` como variable sobre valores.

```
#domain valor(V).
```

Restricción 1. En cada casilla hay exactamente un valor.

```
1 {estado(X,Y,M): valor(M)} 1.
```

Restricción 2. Un número no puede aparecer dos veces en una columna.

```
:- estado(XA,Y,V),
   estado(XB,Y,V),
   XA != XB.
```

Restricción 3. Un número no puede aparecer dos veces en una fila.

```
:- estado(X,YA,V),
   estado(X,YB,V),
   YA != YB.
```

Restricción 4. Un número no puede aparecer dos veces en un cuadrado.

```
:- estado(XA,YA,V),
   estado(XB,YB,V),
   mismo_cuadrado(XA,XB),
   mismo_cuadrado(YA,YB),
   XA != XB,
   YA != YB.
```

Definición 4. `mismo_cuadrado(X,Y)` se verifica si `X` e `Y` se encuentran en el mismo cuadrado.

```
mismo_cuadrado(X,Y) :-
   raiz_de_n(M),
   div(X-1,M) == div(Y-1,M).
```

Definición 5. `raiz_de_n(X)` se verifica si `X` es la raiz cuadrada de `N`.

```
raiz_de_n(X) :-
   X*X == n.
```

Nota 1. Se muestra sólo los valores de `estado`.

```
#hide.
#show estado(X,Y,V).
```

Referencias

[1] C. Baral *Knowledge representation, reasoning and declarative problem solving with answer sets.* Cambridge University Press, 2003.

[2] M. Brain *A simple, declarative Su–Doku solver.*

[3] T. Sryjänen *Lparse 1.0 User's Manual.* Helsinki University, 2000.

[4] I. Niemelä, P. Simons y T. Syrjänen *Smodels: A System for Answer Set Programming.* arXiv.cs.AI/0003033 v1 8 Mar 2000.