Práctica 6

Lógica computacional 2020-2

Resolución de sudokus

Fecha de entrega: **martes 9 de junio de 2020**

1. Introducción

El sudoku es un juego matemático que se inventó a finales de la década de 1970, adquirió popularidad en Japón en la década de 1984 y se dio a conocer en el ámbito internacional en 2005 cuando numerosos periódicos empezaron a publicarlo en su sección de pasatiempos.

El objetivo del sudoku es rellenar una cuadrícula de 9×9 celdas (81 casillas) dividida en subcuadrículas de 3×3 (también llamadas "cajas" o "regiones") con las cifras del 1 al 9 partiendo de algunos números ya dispuestos en algunas de las celdas. Lo que importa, es que sean nueve elementos diferenciados, que no se deben repetir en una misma fila, columna o subcuadrícula.

Un sudoku está bien planteado si la solución es única, algo que el matemático Gary McGuire ha demostrado que no es posible si no hay un mínimo de 17 cifras de pista al principio.

La solución de un sudoku establece la restricción añadida de que no se puede repetir un mismo número en una subcuadrícula.

Un sudoku bien hecho solo puede tener una solución, que es la correcta, para ser considerado sudoku. Es decir, un sudoku tiene solución única. G. Mcguire cree haber probado que un cuadrado latino con menos de 17 pistas necesariamente tendrá múltiples soluciones (si la tiene). Es decir, para ser bien planteado es condición necesaria que un sudoku deba tener al menos 17 pistas.

La construcción de un sudoku puede ser realizada a mano eficientemente predeterminando las posiciones de los números dados y asignándoles valores para realizar un proceso deductivo.

La estrategia para resolver este rompecabezas se puede considerar como la combinación de tres procesos: rastreo, marcado y análisis.

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

Figura 1: Ejemplo de un sudoku.

1.1. Rastreo

Por ejemplo rastreando a lo largo y ancho los sietes localizados en cualquier lugar de la rejilla, el jugador puede eliminar todas las celdas vacías de la esquina superior izquierda que no pueden contener un 7. Esto deja sólo una celda posible.

1.2. Marcado

El rastreo viene a interrumpirse cuando no pueden descubrirse nuevos números. En este punto es necesario centrarse en algún análisis lógico. La mayoría encuentra útil guiar este análisis mediante el marcado de números candidatos en las celdas vacías. Hay dos notaciones populares: subíndices y puntos.

En la notación de subíndice, los números candidatos se escriben en pequeño en las celdas. La desventaja es que los rompecabezas originales se publican en periódicos que habitualmente no dejan demasiado espacio para acomodar más que unos pocos dígitos. Si se usa esta notación, los resolutores crean, a menudo, una copia más grande del rompecabezas y emplean un lápiz afilado.

La segunda notación es un patrón de puntos con un punto en la esquina superior izquierda representando un 1 y un punto en la esquina inferior derecha representando un 9. Esta notación tiene como ventaja que puede usarse en el rompecabezas original. Se requiere destreza para el emplazamiento de los puntos, porque la existencia de puntos desplazados o marcas inadvertidas lleva, inevitablemente, a confusión y no son fáciles de borrar sin añadir más confusión.

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

Figura 2: Resolución del sudoku.

1.3. Análisis

Hay dos aproximaciones principales:

En eliminación, el progreso se realiza mediante la sucesiva eliminación de números candidatos para una o más celdas, hasta dejar solo una elección. Después de lograr cada respuesta, debe realizarse un nuevo rastreo (habitualmente comprobando el efecto del último número). Hay una serie de tácticas de eliminación. Una de las más comunes es el "borrado del candidato no coincidente". Las celdas con idéntica configuración de números candidatos se dice que coinciden si la cantidad de números candidatos en cada una es igual al número de celdas que los contienen. Esta aproximación puede ser desaprobada por puristas lógicos por demasiado ensayo y error pero puede llegar a soluciones claras y rápidamente.

Idealmente, se necesita encontrar una combinación de técnicas que eviten alguno de los inconvenientes de los elementos de arriba. El recuento de regiones, filas y columnas puede resultar aburrido. Escribir números candidatos en celdas vacías puede consumir demasiado tiempo. La aproximación "y-si" puede ser confusa a menos que se sea bien organizado. La intención de la cuestión es encontrar una técnica que minimice el recuento y el marcado.

2. Problema

Genere un programa en Prolog de tal manera que le entreguemos un Sudoku inicial y nos regrese todas las posibles respuestas, ya sea que esta sea única o existan varias posibles soluciones.

Al colocar en el programa la función `sudoku` junto con la representación en forma de lista del sudoku que queramos resolver, el programa debe de respondernos con la o las posibles soluciones. Por ejemplo:

```
?- sudoku( _,6,_,1,7,_,_,5,_,_,_,8,3,_,5,6,_,_,2,_,_,_,_,_,_,_,1,
           8,_,_,4,_,7,_,_,6,_,_,6,_,_,_,3,_,_,7,_,_,9,_,1,_,_,4,
           5,_,9,_,_,_,_,_,_,2,_,_,7,2,_,6,9,_,_,_,4,_,5,_,8,_,7,_).

------------------
|9 6 3|1 7 4|2 5 8|
|1 7 8|3 2 5|6 4 9|
|2 5 4|6 8 9|7 3 1|
------------------
|8 2 1|4 3 7|5 9 6|
|4 9 6|8 5 2|3 1 7|
|7 3 5|9 6 1|8 2 4|
------------------
|5 8 9|7 1 3|4 6 2|
|3 1 7|2 4 6|9 8 5|
|6 4 2|5 9 8|1 7 3|
------------------
true.
```

Evidentemente para que puedan resolver el Sudoku deberán representar el Sudoku que se quiera resolver con la función del mismo nombre que recibe una lista que consta de 81 elementos y generar las funciones y condiciones correspondientes de tal forma que cumplan con las condiciones necesarias para que donde se encuentren las casillas vacías puedan ser llenadas con la solución correspondiente.

También deben de considerar las opciones que crean necesarias para dar las posibles soluciones en caso de que no sean únicas como en el siguiente ejemplo, el cual tiene seis posibles soluciones:

```
?- sudoku( _,6,_,1,7,_,_,5,_,_,_,_,_,3,_,5,6,_,_,2,_,_,_,_,_,_,_,1,
           _,_,_,4,_,7,_,_,6,_,_,6,_,_,_,_,3,_,_,7,_,_,9,_,1,_,_,4,
           5,_,9,_,_,_,_,_,_,2,_,_,7,2,_,6,9,_,_,_,4,_,5,_,8,_,_,_).

------------------
|3 6 4|1 7 9|2 5 8|
|8 7 1|3 2 5|6 4 9|
|2 9 5|6 8 4|7 3 1|
------------------
|9 2 8|4 3 7|5 1 6|
|4 1 6|8 5 2|3 9 7|
|7 5 3|9 6 1|8 2 4|
```

```
------------------
|5 8 9|7 1 3|4 6 2|
|1 3 7|2 4 6|9 8 5|
|6 4 2|5 9 8|1 7 3|
------------------
true ;
------------------
|8 6 3|1 7 4|2 5 9|
|9 7 1|3 2 5|6 4 8|
|2 5 4|6 8 9|7 3 1|
------------------
|1 2 5|4 3 7|8 9 6|
|4 9 6|8 5 2|3 1 7|
|7 3 8|9 6 1|5 2 4|
------------------
|5 8 9|7 1 3|4 6 2|
|3 1 7|2 4 6|9 8 5|
|6 4 2|5 9 8|1 7 3|
------------------
true ;
```

3. Consideraciones y ayuda

Para que puedan realizar de mejor forma esta práctica, es necesario que tomen en cuenta las siguientes consideraciones, si no quieren utilizar la ayuda que se les proporciona, está bien sólo recuerden que todo lo que utilicen y hagan deben de **especificar a detalle qué es, cómo funciona y para qué lo utilizan**.

Para poder imprimir en la pantalla el Sudoku de tal forma que lo podamos ver de manera más ordenada pueden de utilizar la función `write`, así como los saltos de línea `\n`. Por ejemplo, si utilizamos el siguiente código,

```
sol(X) :- write('__________________\n'),
write('|'),write('5'),write(' '),write('6'),write(' '),write('7'),write('|'),
write('8'),write(' '),write('9'),write(' '),write('1'),write('|'),write('2'),
write(' '),write('3'),write(' '),write('4'),write('|\n'),
write('|'),write('5'),write(' '),write('6'),write(' '),write('7'),write('|'),
write('8'),write(' '),write('9'),write(' '),write('1'),write('|'),write('2'),
write(' '),write('3'),write(' '),write('4'),write('|\n').
```

Obtenemos el siguiente resultado:

```
?- sol(X).
------------------
|5 6 7|8 9 1|2 3 4|
|5 6 7|8 9 1|2 3 4|
true.
```

Además de esta consideración es recomendable que utilicen la librería `bounds`, la cual se usa de la siguiente forma:

```
:-use_module(library(bounds)).
```

poniendo dicho código al inicio del archivo.

Esta librería es de gran utilidad para realizar funciones predeterminadas tales como `all_different` y `label` las cuales les serán de mucha utilidad para poder resolver su práctica.

```
?- all_different([1,2,3,4]).
true.
?- all_different([1,2,1,4]).
false.
```

Y la función `label` puede ser usada de la siguiente forma:

```
label([A1, A2, A3, A4, A5, A6])
```

y lo que hace la función `label` es asignar el verdadero valor que tienen las variables, para evitar que asigne los valores que el sistema asume que pueden ser de variables tales como _3576, etc. además de que también evita que coloque puntos suspensivos en colocar la posible respuesta.

Evidentemente será parte de su trabajo el revisar exactamente qué y cómo es que se aplican estas funciones así como implementarlas de manera adecuada en su práctica, si no desean utilizar dichas funciones, no hay problema alguno solo asegurense de que el programa realice lo que debe hacer que es mostar la solución aunque no sea de la forma en que se especificó siempre debe de entregar la solución y ser clara. No deben olvidar que todo el código debe de estar perfectamente documentado y especificar claramente que es lo que hacen todas las funciones así como los procesos que se realizan en cada una de ellas. De no encontrar estos requerimientos en sus códigos se restarán puntos a su práctica.

4. Entrega de la práctica

Esta práctica la pueden entregar en equipos de dos personas, si alguien por cualquier motivo no puede o prefiere trabajar solo de igual forma puede entregarla de manera individual.

También podrán utilizar cualquier función nativa de prolog que consideren necesaria, evidentemente, toda función que utilicen deben explicar qué hace y cómo es que funciona, así como tener todo su código y funciones que realicen bien documentadas.