Conocimiento y Razonamiento Automatizado

Sudoku en Prolog

Grado en Ingeniería Informática
Universidad de Alcalá

Pablo García García
Álvaro Jesús Martínez Parra

23 de marzo de 2023

Índice general

1. Introducción **2**
1.1. Objetivo 2
1.2. Predicados auxiliares 2
1.3. Lista de posibilidades 3

2. Reglas de simplificación **5**
2.1. Regla 0 5
2.2. Regla 1 5
2.3. Regla 2 6
2.4. Regla 3 6

3. Mejoras y detalles **8**
3.1. ASCII Art 8
3.2. Detalles 9

Capítulo 1

Introducción

1.1. Objetivo

Durante esta práctica trataremos sudokus como listas de longitud 81. Cada lugar en la lista se corresponde con una casilla del tablero 9×9. Se inicia con un tablero con números entre uno y nueve donde haya un número y con un punto en el resto.

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

Tabla 1.1: Ejemplo de Sudoku

El objetivo será sustituir los puntos por los posibles valores que podrían encontrarse en dichas casillas, y mediante la ayuda de cuatro reglas, ir acotando estas posibilidades. En algunos casos conseguiremos la resolución del Sudoku.

1.2. Predicados auxiliares

En esta sección detallaremos una serie de predicados que utilizaremos a lo largo del código para poder cumplir los diferentes objetivos:

- `reemplazar/4`: dado el índice de un elemento en una lista, este predicado sustituye dicho elemento por otro dado, y devuelve la lista con el reemplazo hecho.
- `contarApariciones/3`: este predicado cuenta las veces que aparece un elemento en una lista de listas, tomando n apariciones en una sublista como una.

- `contarSemejantes/3`: cuenta cuántas veces aparece una lista en una lista de listas.
- `indicesFila/2`: dado el número de una fila ($f \in [0, 8]$), devuelve una lista con los índices de los elementos de la fila f.
- `indicesColumna/2`: dado el número de una columna ($c \in [0, 8]$), devuelve una lista con los índices de los elementos de la columna c.
- `indicesCuadro/2`: dado el número del índice de inicio de un cuadro s, devuelve una lista con los índices de los elementos del cuadro s.
- `conflictivos/2`: dada la posición de un elemento, devuelve una lista con las posiciones conflictivas[1] de dicho elemento.
- `quitarElementoDeConflictivos/4`: como el propio nombre indica, dado un elemento y sus conflictivos, lo quita de estos.
- `quitarLista/4`: lo explicaremos mejor cuando tratemos la regla 2.
- `darFormato/3`: unifica la lista de listas en una lista con motivo de imprimir el tablero por pantalla.
- `contarNumerosSolucion/2` y `sudokuCompletado/1`: nos serán de utilidad a la hora de determinar si mediante simplificaciones hemos resuelto el Sudoku.
- `cogerElementosIndice/4`: dada una lista de índices, devuelve una lista con los elementos del tablero que hay en dichos índices.
- `fila/3`, `columna/3`, `cuadro/3`: devuelve una lista con los elementos de la fila, columna, o cuadro $i-$ésimo.
- `contarAparicionesRegla3/3` y `quitarListaRegla3/4`: serán de utilidad para construir la regla 3, los explicaremos más adelante.

1.3. Lista de posibilidades

"*Dado un sudoku, que recordemos, es una lista de longitud 81, se debe generar la lista de posibilidades para cada uno de los lugares del sudoku. Es decir, cada uno de los lugares ocupados por un punto debe sustituirse por una lista con todos los posibles números que pueden ir en ese lugar, mientras que los lugares ocupados inicialmente por dígitos no se tocan. La lista de posibilidades, por lo tanto, es una lista de la forma* `[[2,3,...],1,[3,4,..],...,[7,9]]` *también de longitud 81.* "

Un ejemplo de cómo se ilustra esto es lo que vemos en la tabla 1.2. En cursiva se muestran los números calculados. Para calcular las posibilidades de valores que existen en una

[1] En un Sudoku solo puede aparecer un número una vez en una fila, columna, o cuadrado. Las posiciones conflictivas de un elemento serán las formadas por su fila, columna, y cuadro.

2345	*25*	9	6	*23457*	*23578*	*2347*	1	*247*
8	*256*	*2346*	*2345*	*23457*	1	*23467*	9	2467
7	*1256*	*12346*	*2345*	*23459*	*235*	*2346*	*2356*	8
125	3	*1278*	*2458*	6	*2578*	*124789*	*278*	*1247*
2	4	*2678*	1	*237*	9	*23678*	*23678*	5
9	*12567*	*12678*	*23458*	*23457*	*23578*	*1234678*	*23678*	*12467*
123	8	*1237*	9	*23*	*236*	5	4	*1267*
6	*29*	*24*	7	1	*25*	*28*	*28*	3
123	*127*	5	*23*	8	4	*1267*	*267*	9

Tabla 1.2: Ejemplo de *posibilidades*

casilla del Sudoku, se llama al predicado `hacerPosibilidades/2`. Distingue dos casos, si se encuentra con un único número x, las posibilidades de esa casilla serán `[x]`; de lo contrario, se obtiene el conjunto de valores que se encuentran en la fila, columna, y cuadro, y las posibilidades serán la siguiente diferencia de conjuntos:

$$\mathcal{P} = \mathcal{F} - \mathcal{C} - \mathcal{S} - \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$$

Capítulo 2

Reglas de simplificación

2.1. Regla 0

"*Si hay un lugar donde solo cabe un número, lo escribimos en el lugar correspondiente y lo eliminamos de los lugares en los que aparezca de los que son conflictivos.* "

Para realizar esto se llama al predicado `regla0/4`. Este lo que hace es, mediante una lista de índices visitados, apunta los que ya ha revisado, y para el índice actual, comprueba si el elemento en dicha posición es una lista unitaria, y en ese caso se quita ese número de sus conflictivos y se añade a la lista de índices visitados (con la ayuda de `conflictivos/2` y `quitarElementoDeConflictivos/4`). Si se ha realizado una simplificación con esta regla, se vuelve a simplificar desde el índice 0, pero evitando los índices ya visitados. En caso de que no haya un solo número, salta al siguiente.

2.2. Regla 1

"*Si hay un número que aparece en una sola de las listas que aparecen en una fila, columna o cuadro, cambiamos la lista por el número y borramos el número del resto de listas de la fila, columna o cuadro.* "

Para realizar esto se llama al predicado `regla1/3`. Este aplica a cada elemento del tablero el predicado `subregla1/4`. Este predicado "tiene cuatro casos", buscar en la fila, columna, cuadro, o elemento que no podemos simplificar más. En cada uno de ellos con la ayuda de `fila/3`, `columna/3`, `cuadro/3`, `contarApariciones/3`, y `reemplazar/4`, calcula si un elemento aparece solo una vez en su fila, columna, y cuadrado, y en dicho caso sustituye dicha aparición por el número. Con ayuda de la regla 0, borramos el resto de apariciones en su fila, columna, o cuadro.

2.3. Regla 2

"*Si dos números aparecen solos en dos lugares distintos de una fila, columna o cuadro, los borramos del resto de lugares de la fila, columna o cuadro correspondiente.* "

Para cada casilla del tablero, obtenemos las casillas de posibilidades de su fila, columna, y cuadro (mediante la ayuda de `fila/3`, `columna/3`, `cuadro/3`) y buscamos cuáles de estas tienen longitud 2. El objetivo será encontrar sólo dos parejas iguales (con `contarSemejantes/3`) en su fila, columna, o cuadro, y en dicho caso con `quitarLista/4`, eliminamos el resto de apariciones de los números que forman la pareja (sin quitar estas dos parejas). Estudiando más a fondo `quitarLista/4`, distinguimos cuatro casos:

- Dos listas son **iguales** → estamos comparando una de las parejas con la otra, por lo que pasamos al siguiente.
- Dos listas son **diferentes**, y la **longitud** de una de ellas es **menor que 2** → también saltamos al siguiente, pues en esta casilla solo hay una posibilidad, que además será definitiva.
- Dos listas son **diferentes**, y la **longitud** de la que estamos revisando es **mayor** que la longitud de la que se compara → en este caso se eliminan los elementos de la pareja (calculamos una diferencia de conjuntos), y una vez hecho este cálculo, devolvemos las nuevas posibilidades al tablero, y seguimos con el siguiente.
- **No quedan índices** → hemos acabado de revisar el tablero, por lo que lo devolvemos con las modificaciones hechas a lo largo de esta regla.

2.4. Regla 3

"*Si en tres lugares de una fila, columna o cuadro solo aparecen tres números distintos, borramos los números de las restantes listas de la fila, columna o cuadro.* "

Para aplicar la regla tres lo que hacemos es buscar una casilla cuya longitud l sea $1 < l < 4$, de forma que aquellas que tienen longitud 2 o 3, son candidatas a poder ayudar a simplificar otras posibilidades. Una vez se ha encontrado una de estas casillas, con `subregla3/3`, lo que hacemos es buscar en la fila, columna, o cuadro, otras dos casillas tales que al unirlas, sean tres elementos únicos (predicado `contarAparicionesRegla3/3`):

$$\mathcal{T} = C_1 \cup C_2 \cup C_3 = \{e_1, e_2, e_3\}, \text{ con } \begin{array}{rcl} e_1 & \neq & e_2 \\ e_2 & \neq & e_3 \\ e_1 & \neq & e_3 \end{array}$$

Una vez encontrado un caso que satisfaga esta regla, se eliminan los números de esta unión del resto de casillas de la fila, columna, o cuadro donde se haya encontrado esta relación. Si a la hora de simplificar encontramos una casilla con tres posibilidades, ya tenemos el conjunto de tres números que debemos de buscar en la fila, columna, o cuadro. En caso de tener una casilla con dos posibilidades, se realizan combinaciones con esos dos

elementos y otro número diferente a ellos. Esto se hace en `subregla3/4`, y en el predicado `comprobarCombinacionesRegla3/5`, lo que se hace es intentar aplicar la `subregla3/4` a cada conjunto de tres elementos resultante de hacer las combinaciones posibles (como si se hubiera entrado en una casilla con tres posibilidades).

Finalmente, el otro predicado usado es `quitarListaRegla3/4`, que lo que hace es una tarea similar a `quitarLista/4`, pero adaptado a la regla 3.

Capítulo 3

Mejoras y detalles

3.1. ASCII Art

Como mejora de la práctica, hemos propuesto una manera más amigable de ver el Sudoku por consola, usando ASCII art. En esto están involucrados los predicados `darFormato/3`, `imprimirTablero/1`, e `imprimirElemento/2`. El objetivo común de estos predicados es ir imprimiendo por pantalla los elementos del Sudoku, y en función de las posiciones de estos, añadir caracteres ASCII que adornen el puzzle. Veamos dos ejemplos, que se corresponden con la visualización final de un ejemplo que puede resolverse aplicando las reglas solicitadas (Sudoku 1), y otro que no puede llegar a ser resuelto (Sudoku 18):

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

(a) Sudoku 1 (inicio)

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

(b) Sudoku 1 (final)

Figura 3.1: ASCII Art Sudoku 1

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

(a) Sudoku 18 (inicio)

```
+-------+-------+-------+
| [3,8] [5] [7] | [4] [1] [6,8] | [2] [3,6,8] [9] |
| [4] [9] [1] | [2,6,7,8] [2,6,7,8] [3] | [6,7,8] [6,8] [5] |
| [3,8] [6] [2] | [7,8] [9] [5] | [1,7,8] [1,3,8] [4] |
+-------+-------+-------+
| [1,6,7,9] [1,2,7,8] [8,9] | [2,3,6,7,8,9] [2,3,6,7,8] [4] | [1,7,8,9] [5] [1,7,8] |
| [1,5,6,7,9] [1,2,7,8] [4] | [2,5,6,7,8,9] [2,5,6,7,8] [6,7,8,9] | [3] [1,8,9] [1,7,8] |
| [5,7,9] [7,8] [3] | [1] [5,7,8] [7,8,9] | [4] [2] [6] |
+-------+-------+-------+
| [1,9] [1,4,8] [6] | [5,8,9] [4,5,8] [2] | [1,8,9] [7] [3] |
| [2] [1,4,7,8] [8,9] | [3,6,7,8,9] [3,4,6,7,8] [6,7,8,9] | [5] [1,6,8,9] [1,8] |
| [7,9] [3] [5] | [6,7,8,9] [6,7,8] [1] | [6,8,9] [4] [2] |
+-------+-------+-------+
```

(b) Sudoku 18 (final)

Figura 3.2: ASCII Art Sudoku 18

3.2. Detalles

Se han cubierto todos los objetivos solicitados, funcionando correctamente las cuatro reglas, sin ningún error de implementación. Se han hecho diversas pruebas con los Sudokus proporcionados, pudiendo resolverse los Sudokus (1, 2, 3, 4, 5, 6, 7, 8, 9, 13, 15, 21, 23), y quedando simplificados (10, 11, 12, 14, 16, 17, 18, 19, 20, 22, 24, 25).

Para realizar la consulta a la simplificación del Sudoku, se deberá consultar el predicado: `sudoku(X)`, siendo `X` la lista que representa al Sudoku inicial, obtenidos del archivo `sudokus_pruebas.pl`.

Respecto al reparto de tareas, la práctica se ha realizado por ambos integrantes del grupo a la vez en una llamada de Discord, aportando cada uno soluciones y alternativas a los problemas que se iban encontrando.

Para finalizar, las referencias consultadas han sido, los apuntes de la asignatura, el manual de SWI Prolog, y StackOverFlow.