Grado en Ingeniería Informática

Universidad de Alcalá

Conocimiento y Razonamiento Automatizado

PRACTICA 1: SUDOKU EN PROLOG

Dario Reimundez Cecilia

14 de marzo de 2023

Índice

PARTE 1

INTRODUCCION

1.1 Inicio y fundamentos

Un sudoku está compuesto por 9 filas y 9 columnas, que se dividen en 9 cuadrantes de 3x3. En esta práctica se partirá de una lista de 81 elementos que representara el tablero que contiene el sudoku.

Como sugiere el enunciado, las casillas que no contengan un número, se rellanara con un ".", indicando que todavía no está encontrado un número que corresponde a esas coordenadas.

5	3	4	.	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

Imagen 1.1: Ejemplo tablero

1.2 Objetivo y metodología

A través de un lenguaje de programación lógica basado en predicados, conocido como "Prolog". Se espera poder desarrollar un programa que sea capaz de resolver diferentes tableros de sudoku propuestos.

La práctica busca orientarnos para comprender principios fundamentales de la programación lógica, especialmente cómo se modelan los problemas y se resuelven mediante reglas y relaciones lógicas en Prolog.

Como veremos en los siguientes apartados, se seguirán las directrices propuestas por el enunciado. Como primera aproximación deberemos de crear la lista de posibilidades de cada casilla para, más adelante, crear las reglas que trabajen sobre estas.

1.3 Predicados “dinámicos”

Por las propias limitaciones de los predicados, resulta imprescindible el uso de predicados dinámicos. En nuestra implementación, declaramos cinco de estos que nos permiten crear, borrar y modificar predicados en tiempo de ejecución.

```
dynamic existe/3.
dynamic fila/2.
dynamic columna/2.
dynamic cuadrante/3.
dynamic posibilidades/3
```

Imagen 1.2: Predicados declarados como dinámicos

Más tarde, veremos que, por cómo hemos implementado las reglas, resulta redundante definir cada predicado. En su lugar, explicamos algunos de los más importantes cuando expliquemos las reglas.

1.4 Cálculo de posibilidades

Deberemos de crear una lista para cada casilla. Esta lista contendrá los posibles valores que pueden ocuparla cumpliendo las reglas del sudoku. Estas reglas definen que en una misma fila, columna y cuadrante no puede repetirse los números.

Para ello, identificaremos los números (sin repeticiones) que conforman la lista, columna y cuadrante de cada casilla y excluiremos estos números del conjunto completo del 1 al 9. Así, obtendremos la lista de posibilidades que necesitaremos para aplicar las reglas más adelante.

```
% Busca las posibilidades de la casilla que quieras
posibilidades(X, Y, Posibilidades) :-
     ocupados(X, Y, Ocupados),
     numeros(Numeros),
     contrario(Ocupados, Numeros, Posibilidades).    % Devuelve los numeros que faltan en la lista

% Utiliza buscarNumeros solo si no existe numero en la casilla
ocupados(X, Y, Ocupados) :-
    (not(existe(_, X, Y)) ->
    buscarNumeros(X, Y, Ocupados)).

% Busca los nÃºmeros que ya estÃ¡n en la fila, columna y cuadrante
buscarNumeros(X, Y, Numeros) :-
    fila(Y, Fila),                      % Busca los nÃºmeros de la fila
    columna(X, Columna),                % Busca los nÃºmeros de la columna
    localCuadr(X, Y, Cuadrante),        % Busca los nÃºmeros del cuadrante
    merge(Fila, Columna, Temp),         % Une los nÃºmeros de la fila y columna
    merge(Temp, Cuadrante, Numeros).    % Une los nÃºmeros de la fila, columna y cuadrante

% Devuelve los nÃºmeros de la lista Ys que faltan en la lista Xs con output en Zs
contrario([], [], []).
contrario(_, [], []).
contrario([], [Y|Ys], [Y|Ys]).
contrario([X|Xs], [Y|Ys] , Zs) :-
    X @< Y,
    contrario( Xs, [Y|Ys], Zs ).
contrario([X|Xs], [Y|Ys], [Y|Zs]) :-
    X @> Y,
    contrario( [X|Xs], Ys, Zs).
contrario([X|Xs], [Y|Ys], Zs) :-
    X == Y,
    contrario(Xs, Ys, Zs).
```

Imagen 1.3: Código para cálculo de posibilidades

Como ejemplo de implementación podemos comprobar en *Imagen 1.3* como “posibilidades(...)”, busca los números ocupados, obtiene los números del 1 al 9 y devuelve las posibilidades.

En concreto, si obtenemos las posibilidades no quedaría una tabla donde cada casilla tiene la lista de posibilidades.

PARTE 2

REGLAS DE SIMPLIFICACIÓN

2.1 Regla 0

El principal predicado encargado de esta tarea es **regla0/0**. Este simplemente utiliza otros predicados más complejos como **aplicarRegla0/1** (usa **detectarNumeroUnico/2** para cada casilla) y **actualizarPosibilidades/0** (borra todas las posibilidades y las calcula de nuevo). A su vez, **detectarNumeroUnico/2** y evalúa si solo existe una posibilidad en la lista de posibilidades. Si es así, crea un nuevo predicado para indicar existe ese número en la casilla.

2.2 Regla 1

Vemos que regla1/0 se encarga de llamar a regla1Fila/1, regla1Cuadrante/2 y regla1Columna/1 de manera un poco rudimentaria pero que ayuda a visualizar que se está haciendo. Como la función de estos tres últimos predicados mencionados es muy similar, procedemos a explicar solo uno de ellos.

Por ejemplo, regla1Fila/1 que deriva en otros predicados para hallar el resultado. Por la manera en que lo hemos implementado hay muy predicados, es muy complejo detallar cuidadosamente cada uno. Los dos más importantes implicados son buscarNumeroFila/4, al que indicas un numero especifico y devuelve cuantas veces lo ha encontrado y buscarNumerosFila/3, que lo hace con cada número y devuelve una lista con las veces que ha encontrado cada uno. Luego, buscarRegla1/3, devuelve los números que salen solo una vez y aplicarRegla1Fila/3 aplica la regla uno en la fila.

2.3 Regla 2

En este caso podemos explicar la regla para el cuadrante para que se entienda. Primero, el predicado **parPosibilidadesCuadrante/2** calcula dónde existen pares de posibilidades únicas en un cuadrante específico, guardando esta

información en una estructura llamada **pIC/3**que almacena las coordenadas y la lista de posibilidades únicas **Res**.

Luego, **aplicarParPosibilidadesCuadrante/2** busca y aplica la regla 2 para la posición en las coordenadas dentro del cuadrante. Si encuentra un par de posibilidades únicas en esa posición y las posibilidades son únicas en esa casilla, entonces elimina esas posibilidades del resto de las casillas en el cuadrante.

El predicado **cambiarPosibilidadesCuadranteDuo/4** se encarga de realizar los cambios necesarios en las posibilidades de todas las casillas del cuadrante, eliminando las posibilidades correspondientes a los números que forman el par único.

Finalmente, **regla2CuadranteDef/0** coordina la aplicación de la regla 2 para todos los cuadrantes del Sudoku. Utiliza los predicados anteriores para buscar y eliminar los pares de posibilidades únicas en cada cuadrante, asegurando que la regla 2 se aplique correctamente en todo el tablero del Sudoku.

2.4 Regla 3

Primero, se tienen los predicados **aplicarTrioPosibilidadesColumna/2**, **aplicarTrioPosibilidadesFila/2** y **aplicarTrioPosibilidadesCuadrante/2**, los cuales buscan tríos de posibilidades únicas en cada columna, fila y cuadrante, respectivamente. Si encuentran un trío de posibilidades únicas en una casilla y las posibilidades son únicas en esa casilla, se eliminan esas posibilidades del resto de las casillas en la fila, columna o cuadrante correspondiente.

Luego, los predicados **aplicarRegla3Columna/1**, **aplicarRegla3Fila/1** y **aplicarRegla3Cuadrante/1** coordinan la aplicación de la regla 3 para todas las columnas, filas y cuadrantes del Sudoku, respectivamente. Utilizan los predicados anteriores para buscar y eliminar los tríos de posibilidades únicas en cada columna, fila y cuadrante, asegurando que la regla 3 se aplique correctamente en todo el tablero del Sudoku.

Finalmente, los predicados **regla3ColumnaDef/0**, **regla3FilaDef/0** **regla3CuadranteDef/**0 coordinan la aplicación de la regla 3 para todas las columnas, filas y cuadrantes del Sudoku, respectivamente. Luego, el predicado **regla3/0** se encarga de ejecutar estos procesos para todas las partes del tablero, lo que garantiza que se aplique la regla 3 en todas las áreas del Sudoku y finalmente se imprime el tablero resultante.