Programación Lógica
Laboratorio 2 - Informe

Javier Morales	Javier Pérez	Melisa Techera
4.780.020-9	4.697.236-2	5.138.335-4

June 15, 2018

1 Estructura

Se implementó un módulo kakuro.pl que consulta tres archivos externos parte2.pl, parte2clp.pl y parte3.pl para hacer más manejable el desarrollo. El archivo principal contiene la implementación de la parte 1 del obligatorio.

2 Glosario

En el código y en este informe se utiliza el concepto de *bloque*, en sus variantes de fila y columna, para designar al subconjunto de una fila o columna afectado por una directiva de suma.

3 Parte 1

Los predicados principales de esta parte fueron provistos por la letra. Se implementaron también predicados auxiliares que se encuentran comentados en el código.

Para el manejo del estado del tablero, necesario para detectar errores, se agregó un tercer componente a la tupla *estado* del esqueleto de código provisto. La misma almacena el estado actual del tablero. Esta decisión se tomó por los siguientes motivos:

- La detección de errores requiere conocer qué números están ya ingresados en el tablero. Para almacenar esta información se utiliza esta variable, de forma independiente al módulo de gráficos.
- El uso del parámetro *Tablero* ya existente no es viable debido a que el usuario puede cambiar los valores de un número, o incluso borrarlos, y la naturaleza de prolog no permite des-asignar variables.

El uso de esta estrategia hace necesarios dos parámetros en ciertos predicados, uno referente al Tablero previo, y otro al Tablero posterior (análogamente a cómo se maneja la tupla estado).

A nivel gráfico, los errores se muestran (con borde rojo) en la casilla que al rellenarse causó el error. (por ejemplo, si hay múltiples instancias del mismo número en una fila o columna se muestra en rojo el segundo al momento de rellenarse). Si a una fila que ya tiene errores se le colocan nuevos números, estos también se marcan en rojo.

4 Parte 2

4.1 Prolog Estándar

La resolución mediante prolog estándar tiene como predicados principales generarFilas y chequearColumnas, siguiendo el esquema de generación y chequeo.

La generación de filas coloca dígitos en las variables y realiza un chequeo inicial para incrementar eficiencia, asegurándose que sean distintos y sumen los valores necesarios.

El chequeo de columnas se asegura de la unicidad de dígitos en los bloques y las correctitud de las sumas en las columnas.

4.2 CLPFD

La resolución mediante CLPFD fue adaptada desde la estándar, con una estructura similar. Los predicados más importantes son generarFilasCLP y generarColumnasCLP.

La diferencia principal a nivel de implementación es que la generación es más declarativa e incluye los chequeos embebidos en las restricciones. Luego, solo resta realizar labeling a las variables.

4.3 Eficiencia

Se realizaron pruebas de eficiencia para la parte 2 en sus dos implementaciones. Se muestra el resultado de tiempo hasta encontrar el primer resultado, si bien al pedir más resultados la ejecución continúa y devuelve *false* pues no existen más soluciones.

Vars denota la cantidad de casilleros blancos en cada tablero.

Puede observarse que prolog estándar no logra resolver los tableros de dimensión mayor, mientras que CLP los resuelve en tiempo despreciable. Esto se debe a la naturaleza exponencial de las permutaciones que se generan en el modo estándar, pero son resueltas mediante restricciones en CLP.

Todas las pruebas realizadas en una de las computadoras de facultad (Sala UDELAR D).

Table 1: Eficiencia

Nº Juego	**Dimensión**	**Vars**	**STD**	**CLP**
1	6x5	14	4.93s	0.00s
2	5x5	10	0.000s	0.00s
3	9x8	36	>300s	0.00s
4	5x5	12	11.32s	0.00s
5	5x5	12	1.62s	0.00s
6	9x8	28	>300s	0.00s

5 Parte 3

La generación de tableros se dividió en los siguientes procedimientos principales:

- **bloques:** instancia p(_,_) en una matriz de variables toda la primer fila, la primer columna, y celdas aleatorias del interior con probabilidad 1/10.
- **restricciones:** coloca restricciones clpfd en las celdas vacías y en las variables de las celdas p. Las celdas p con valor adyacente a derecha otro p o vacío, su valor correspondiente a la suma fila es cero. Análogamente para los valores p adyacentes hacia abajo y la suma de columna.
- **labelear:** realiza labeling de las variables sujetas a restricciones de manera aleatoria. Se utilizó $random_between$, $random_variable$ y $random_value$.
- **simplificar:** convierte los p que tienen valores cero a f, c y n según corresponda.