UNIVERSIDAD DE GRANADA

3º CSI 2019/20 - Grupo 1

Grado en Ingeniería Informática

Práctica 2: Satisfacción de restricciones

Autor:
Antonio David Villegas Yeguas
DNI:
77021623-M

Asignatura:
Técnicas de los Sistemas Inteligentes
Correo:
advy99@correo.ugr.es

1 de mayo de 2020

1. Introducción

En esta práctica se nos propone resolver distintos problemas de satisfacción de restricciones. En cada problema, dado su enunciado, hay que decidir cuales son las variables que intervienen en el problema, el dominio que tienen dichas variables, y que restricciones tienen esas variables, como bien sabemos tras estudiar el tema 2 de la teoría de la asignatura.

Para resolver estos problemas propuestos usaremos la herramienta MiniZinc, un lenguaje de modelado de restricciones que cuenta con un IDE con el que podremos representar los problemas y resolverlos. Este lenguaje cuenta con distintas herramientas para resolver los problemas de satisfacción de restricciones (*CSP solvers*), aunque para esta práctica solo utilizaremos Gecode, instalado por defecto.

2. Ejercicio 1

Puzzle Cripto-aritmético. El siguiente problema plantea un problema criptoaritmético, de forma que cada letra codifica un único dígito (es decir, un número entero en [0,9]) y cada dígito está asignado a una única letra. Se pide encontrar una asignación de dígitos a letras que satisfaga la siguiente suma:

```
    TESTE
+   FESTE
+   DEINE
 =========
   KRAFTE
```

2.1. Variables

Tendremos 10 variables, una por letra. En MiniZinc lo representaremos con un array de 10 posiciones.

2.2. Dominio de las variables

Cada variable es un entero entre el número 0 y el 9 (ambos incluidos).

Usaremos variables auxiliares para gestionar los acarreos.

2.3. Restricciones

1: Cada variable tiene que tomar un valor distinto. Esto lo conseguiremos con el uso de la función `alldifferent(array)` de MiniZinc.

2: La suma de cada columna tiene que coincidir (con su respectivo acarreo). Usaremos 5 variables para almacenar el acarreo de cada suma, y haremos uso de `constraint` para hacer que las sumas sumando el acarreo (y dividido por 10, para que el resultado este entre 0..9) sean igual a la letra correspondiente. El acarreo será la misma operación, pero aplicando la operación módulo 10.

2.4. Resultado

```
Compiling ej1.mzn
Running ej1.mzn
Solución:
        E: 0
        T: 3
        N: 7
        S: 8
        I: 9
        F: 6
        A: 2
        D: 5
        R: 4
        K: 1

          TESTE              30830
        + FESTE            + 60830
        + DEINE            + 50970
        ========           ========
         KRAFTE             142630
----------
Finished in 124msec
```

Figura 1: Solución obtenida para el ejercicio 1.

3. Ejercicio 2

Encontrar un número X de 10 dígitos que satisfaga que el primer dígito de X representa el número de 0s en X, el segundo dígito de X representa el número de 1s en X, etc...

Por ejemplo, el número X=6210001000 satisface dicha condición.

3.1. Variables

Usaremos un array con 10 posiciones, de la 0 a la 9 (ambos incluidos) para representar el número X.

3.2. Dominio de las variables

Las variables son números enteros de 0 a 9.

3.3. Restricciones

El indice del dígito equivale al número de veces que aparece el dígito en el número X. Esto lo conseguimos en MiniZinc con un bucle, en el que para todo i desde 0 hasta 9, tiene que aparecer i veces.

3.4. Resultado

```
Finished in 124msec
Compiling ej2.mzn
Running ej2.mzn
X = [6, 2, 1, 0, 0, 0, 1, 0, 0, 0]
----------
Finished in 126msec
```

Figura 2: Solución obtenida para el ejercicio 2.

4. Ejercicio 3

Encontrar una asignación de horarios que satisfaga las siguientes condiciones (condiciones en la sección de restricciones).

4.1. Variables

Usaremos un array con tantos elementos como profesores.

También usaremos una matriz con tantas filas como profesores y dos columnas, para representar el inicio y fin del horario de cada profesor.

4.2. Dominio de las variables

El array para la asignación de profesores puede tomar valores entre 9 y 14, las horas donde puede comenzar una clase, es decir, si, por ejemplo, toma el valor 9 en la posición 2, quiere decir que el profesor 2 da clase de 9:00 a 10:00.

La matriz puede tomar valores entre 9 y 15, el horario dado.

4.3. Restricciones

Las restricciones son que un aula solo puede estar ocupada por un profesor, esto lo hemos conseguido al usar una array con una posición por hora.

Otra restricción a tener en cuenta es que las clases son de 1 hora, luego no se repiten ningún profesor, usaremos `alldifferent(profesores)` para tener en cuenta esta restricción.

Cada profesor tiene un horario disponible. Esto lo solucionamos haciendo uso de la matriz de horarios, asegurando que la posición de un profesor en el array es mayor o igual que la hora de inicio y menor que la hora de fin (en la hora de fin menor estricto, ya que si un horario va de 9:00 a 13:00, no se puede dar clase a las 13:00 - 14:00).

4.4. Resultado

```
Finished in 126msec
Compiling ej3.mzn
Running ej3.mzn
Profesor 1:
        Horario disponible: 11 - 15
        Horario asignado : 14 - 15
Profesor 2:
        Horario disponible: 11 - 13
        Horario asignado : 12 - 13
Profesor 3:
        Horario disponible: 10 - 14
        Horario asignado : 13 - 14
Profesor 4:
        Horario disponible: 10 - 13
        Horario asignado : 10 - 11
Profesor 5:
        Horario disponible: 11 - 13
        Horario asignado : 11 - 12
Profesor 6:
        Horario disponible: 9 - 15
        Horario asignado : 9 - 10
----------
Finished in 140msec
```

Figura 3: Solución obtenida para el ejercicio 3.

5. Ejercicio 4

5.1. Variables

5.2. Dominio de las variables

5.3. Restricciones

5.4. Resultado

```
Compiling ej4.mzn
Running ej4.mzn
9:00:
[0]  [8]  [6]  [2]

10:00 :
[11]  [0]  [7]  [1]

11:00 :
[10]  [9]  [4]  [3]

12:00 :
[12]  [5]  [0]  [0]

 Asignaciones: Nada = 0, IA1 = 1, TSI1 = 2, FBD1 = 3, IA2 = 4, TSI2 = 5, FBD2 = 6, IA3 = 7, TSI3 = 8, FBD3 = 9, IA4 = 10, TSI4 = 11, FBD4 = 12,
Clases del profesor 1 : IA1, IA2, TSI1, TSI2
Clases del profesor 2 : FBD1, FBD2
Clases del profesor 3 : TSI3, TSI4, FBD3, FBD4
Clases del profesor 4 : IA3, IA4
----------
Finished in 157msec
```

Figura 4: Solución obtenida para el ejercicio 4.

6. Ejercicio 5

6.1. Variables

6.2. Dominio de las variables

6.3. Restricciones

6.4. Resultado

```
Compiling ej5.mzn
Running ej5.mzn
Lunes:
        [6, 4, 4, 0, 3, 3]
Martes:
        [4, 4, 2, 0, 1, 1]
Miercoles:
        [6, 5, 5, 0, 3, 3]
Jueves:
        [7, 5, 5, 0, 1, 1]
Viernes:
        [7, 8, 8, 0, 2, 9]
----------
Finished in 5s 840msec
```

Figura 5: Solución obtenida para el ejercicio 5.

7. Ejercicio 6

7.1. Variables

7.2. Dominio de las variables

7.3. Restricciones

7.4. Resultado

```
Compiling ej6.mzn
Running ej6.mzn
Nacionalidades: [andaluz, navarro, vasco, catalan, gallego]
Bebidas       : [agua, te, leche, zumo, cafe]
Profesion     : [diplomatico, medico, escultor, violinista, pintor]
Mascota       : [zorro, caballo, caracoles, perro, cebra]
Color de casa : [amarillo, azul, rojo, blanco, verde]

La cebra esta en la casa 5, donde vive el gallego, que bebe cafe, trabaja en pintor y tiene la casa pintada de verde
La persona que bebe agua vive en la casa 1, es andaluz, tiene como mascota zorro, trabaja en diplomatico y tiene la casa pintada de amarillo
----------
Finished in 150msec
```

Figura 6: Solución obtenida para el ejercicio 6.

8. Ejercicio 7

8.1. Variables

8.2. Dominio de las variables

8.3. Restricciones

8.4. Resultado

```
Compiling ej7.mzn
Running ej7.mzn
Orden (si no aparece un número, por ejemplo el 3, es porque se esta realizando en paralelo con otra tarea) :
[1, 2, 5, 2, 6, 6, 6, 2, 9]
Coste: 18
----------
==========
Finished in 128msec
```

Figura 7: Solución obtenida para el ejercicio 7.

9. Ejercicio 10

9.1. Variables

9.2. Dominio de las variables

9.3. Restricciones

9.4. Resultado

```
seleccionados = array1d(1..12, [true, true, true, true, true, false, false, false, false, false, false, false]);
peso_actual = 240;
preferencia = 605;
----------
seleccionados = array1d(1..12, [true, false, true, true, true, false, true, false, false, false, false, false]);
peso_actual = 254;
preferencia = 630;
----------
seleccionados = array1d(1..12, [true, true, true, true, true, false, true, false, false, false, false, false]);
peso_actual = 267;
preferencia = 665;
----------
seleccionados = array1d(1..12, [true, true, true, true, true, false, false, false, false, false, true, false]);
peso_actual = 251;
preferencia = 675;
----------
seleccionados = array1d(1..12, [true, false, true, true, true, false, true, false, false, false, true, false]);
peso_actual = 265;
preferencia = 700;
----------
seleccionados = array1d(1..12, [true, true, true, true, true, false, false, false, true, false, true, false]);
peso_actual = 274;
preferencia = 705;
----------
==========
Finished in 169msec
```

Figura 8: Solución obtenida para el ejercicio 10.