Universidad del Valle
Escuela de Ingeniería de Sistemas y Computación
Programación por Restricciones
Desarrollo: Taller Modelamiento e Implementación CSPs

Juan Marcos Caicedo Mejía (1730504-3743)

2020

PARTE 1 (CSPs)

1. Sudoku

- Variables:

  - `subcuadricula`

    Representa la dimensión de cada subcuadricula. En un Sudoku común y corriente esto tiene un valor de 3.

  - `cuadricula`

    Representa la dimensión de toda la cuadricula grande (la que contiene las subcuadriculas). Al igual que la variable anterior, en un Sudoku común esta variable tiene el valor de 9.

  - `entrada`

    Representa el tablero de Sudoku inicial que posee algunos números preestablecidos. En esta matriz, el 0 representa una celda vacía.

  - `salida`

    Representa el tablero variable sobre el cual las restricciones tomarán acción y construirán la solución del Sudoku.

  - `subcuadricula_uno`

    Es un array que contiene los 9 números correspondientes a la primera subcuadricula

  - `subcuadricula_dos`

    Es un array que contiene los 9 números correspondientes a la segunda subcuadricula

  - `subcuadricula_tres`

    Es un array que contiene los 9 números correspondientes a la tercera subcuadricula

  - `subcuadricula_cuatro`

    Es un array que contiene los 9 números correspondientes a la cuarta subcuadricula

  - `subcuadricula_cinco`

    Es un array que contiene los 9 números correspondientes a la quinta subcuadricula

- `subcuadricula_seis`

  Es un array que contiene los 9 números correspondientes a la sexta subcuadricula

- `subcuadricula_siete`

  Es un array que contiene los 9 números correspondientes a la séptima subcuadricula

- `subcuadricula_ocho`

  Es un array que contiene los 9 números correspondientes a la octava subcuadricula

- `subcuadricula_nueve`

  Es un array que contiene los 9 números correspondientes a la novena subcuadricula

- Dominios:

  - `dimension_cuadricula = 1..cuadricula`

    Este dominio es útil para poder acceder a todas las celdas de la cuadrícula grande, pues es el que demarca que los números que estén en el tablero solo sean del 1 al 9.

- Restricciones:

  - Una primera restricción que solo sirve para poder copiar los datos del tablero inicial del Sudoku consiste en que si alguna celda en el tablero de la variable de entrada contiene un número mayor que 0, este mismo número se copie en la misma ubicación en el tablero de salida, si no es así, es decir, hay una celda con 0, se omite su valor:

$$\forall i, j \in \{1, 9\} \mid entrada(i, j) > 0 \therefore salida(i, j) = entrada(i, j)$$

  - Luego, una restricción importante es para garantizar que los números de todas las filas sean distintos:

$$\forall i, j \in \{1, 9\} \mid$$

$$x_{1,1} \neq x_{1,2} \neq x_{1,3} \neq ... \neq x_{1,9}$$

$$\wedge$$

$$x_{2,1} \neq x_{2,2} \neq x_{2,3} \neq ... \neq x_{2,9}$$

$$\wedge$$

$$...$$

$$x_{9,1} \neq x_{9,2} \neq x_{9,3} \neq ... \neq x_{9,9}$$

- Luego, una restricción importante es para garantizar que los números de todas las columnas sean distintos:

$$\forall i, j \in \{1, 9\} \mid$$

$$x_{1,1} \neq x_{2,1} \neq x_{3,1} \neq ... \neq x_{9,1}$$

$$\wedge$$

$$x_{1,2} \neq x_{2,2} \neq x_{3,2} \neq ... \neq x_{2,9}$$

$$\wedge$$

$$...$$

$$x_{1,9} \neq x_{2,9} \neq x_{3,9} \neq ... \neq x_{9,9}$$

  Donde $x_{i,j}$ es el número de la celda en la posición (i, j).

- Luego, usando las variables de:

  `subcuadricula_uno`

  hasta

  `subcuadricula_nueve`

  Se impusieron las restricciones que permiten que en cada subcuadrícula no se repitan los números del 1 al 9. Por cada variable de subcuadricula (que contiene los numeros de dicha subcuadricula) se impuso la restricción de

  `alldifferent`

  Para la

  `subcuadricula_uno`

La restricción sería así:

$$x_{1,1} \neq x_{1,2} \neq x_{1,3}$$
$$\wedge$$
$$x_{2,1} \neq x_{2,2} \neq x_{2,3}$$
$$\wedge$$
$$x_{3,1} \neq x_{3,2} \neq x_{3,3}$$

Para la

`subcuadricula_dos`

La restricción sería así:

$$x_{1,4} \neq x_{1,5} \neq x_{1,6}$$
$$\wedge$$
$$x_{2,4} \neq x_{2,5} \neq x_{2,6}$$
$$\wedge$$
$$x_{3,4} \neq x_{3,5} \neq x_{3,6}$$

Para la

`subcuadricula_tres`

La restricción sería así:

$$x_{1,7} \neq x_{1,8} \neq x_{1,9}$$
$$\wedge$$
$$x_{2,7} \neq x_{2,8} \neq x_{2,9}$$
$$\wedge$$
$$x_{3,7} \neq x_{3,8} \neq x_{3,9}$$

Para la

`subcuadricula_cuatro`

La restricción sería así:

$$x_{4,1} \neq x_{4,2} \neq x_{4,3}$$
$$\wedge$$
$$x_{5,1} \neq x_{5,2} \neq x_{5,3}$$
$$\wedge$$
$$x_{6,1} \neq x_{6,2} \neq x_{6,3}$$

Para la

`subcuadricula_cinco`

La restricción sería así:

$$x_{4,4} \neq x_{4,5} \neq x_{4,6}$$
$$\wedge$$
$$x_{5,4} \neq x_{5,5} \neq x_{5,6}$$
$$\wedge$$
$$x_{6,4} \neq x_{6,5} \neq x_{6,6}$$

Para la

`subcuadricula_seis`

La restricción sería así:

$$x_{4,7} \neq x_{4,8} \neq x_{4,9}$$
$$\wedge$$
$$x_{5,7} \neq x_{5,8} \neq x_{5,9}$$
$$\wedge$$
$$x_{6,7} \neq x_{6,8} \neq x_{6,9}$$

Para la

`subcuadricula_siete`

La restricción sería así:

$$x_{7,1} \neq x_{7,2} \neq x_{7,3}$$
$$\wedge$$
$$x_{8,1} \neq x_{8,2} \neq x_{8,3}$$
$$\wedge$$
$$x_{9,1} \neq x_{9,2} \neq x_{9,3}$$

Para la

`subcuadricula_ocho`

La restricción sería así:

$$x_{7,4} \neq x_{7,5} \neq x_{7,6}$$
$$\wedge$$
$$x_{8,4} \neq x_{8,5} \neq x_{8,6}$$
$$\wedge$$
$$x_{9,4} \neq x_{9,5} \neq x_{9,6}$$

Para la

`subcuadricula_nueve`

La restricción sería así:

$$x_{7,7} \neq x_{7,8} \neq x_{7,9}$$
$$\wedge$$
$$x_{8,7} \neq x_{8,8} \neq x_{8,9}$$
$$\wedge$$
$$x_{9,7} \neq x_{9,8} \neq x_{9,9}$$

2. Kakuro

- Variables:
  - `ancho`
    Constante entera que se encarga de definir cuál será el ancho (en cuadrículas) del Kakuro.
  - `alto`
    Constante entera que se encarga de definir cuál será el alto (en cuadrículas) del Kakuro.
  - `entrada_casillas`
    Matriz constante que hace parte del input del Kakuro: esta matriz denota las casillas disponibles para rellenar (casillas blancas) como 0, y las casillas inhabilitadas para rellenar (generalmente las grises o que tienen diagonal) como -1.
  - `entrada_casillas_left`
    Matriz constante que hace parte del input del Kakuro: esta matriz denota las sumas horizontales en su respectiva casilla a la que pertenecen en el Kakuro como tal.
  - `entrada_casillas_up`
    Matriz constante que hace parte del input del Kakuro: esta matriz denota las sumas verticales en su respectiva casilla a la que pertenecen en el Kakuro como tal.
  - `salida_casillas`
    Matriz de variables de decisión que funciona como apoyo: pues sirve para decirle en qué casillas puede poner valores y en qué no, en la matriz en la cual si funcionan las restricciones.
  - `salida_casillas_def`
    Matriz de variables de decisión que hace el papel del resultado del Kakuro: sobre esta matriz se imponen las verdaderas restricciones de las reglas del Kakuro.
- Dominios:
  - `dimension_ancho = 1..ancho`
    Dominio que denota desde 1 hasta cuánto van las cuadrículas de todo el ancho del Kakuro.

- `dimension_alto = 1..alto`

  Dominio que denota desde 1 hasta cuánto van las cuadrículas de todo el ancho del Kakuro.

- Restricciones:
  - a

3. Secuencia Mágica

- Variables:
  - `n`

    La variable (ingresada por el usuario mediante el IDE o proveída por un archivo de datos .dzn), representa la longitud de la secuencia mágica.
- Dominios:
  - `dominio = 0..n-1`

    Este dominio caracteriza dos cosas: el tamaño de la secuencia mágica (el tamaño del arreglo que la representa) y los posibles valores que toman los números dentro de la secuencia mágica.
- Restricciones:
  - Llamemos *ocurre* a un predicado que recibe una lista l y un número x, así, $ocurre(l, x)$ arroja el número de ocurrencias del número x en la lista l. Otro predicado puede ser *posicion* que dada una lista l y un número x, retorna la posición del número x en la lista l (indexando desde 0). La restricción principal del problema consiste en que la secuencia mágica se caracteriza porque el número i ocurre exactamente x_i veces en la secuencia. Así, la restricción podría modelarse:

    $$\forall x \in l, ocurre(l, posicion(l, x)) = x$$

    Esto quiere decir que el número de la posición de x en la lista l debe ocurrir x veces en la lista l.
  - Una restricción redundante adicional, dicta que la suma de todos los números en la secuencia debe sumar el número n (longitud de la secuencia):

    $$\sum_{i=0}^{n-1} x_i = n$$

    Donde x_i es el i-ésimo elemento de la secuencia mágica.

- Otra restricción adicional, dice que la suma de cada elemento de la secuencia multiplicada por $i - 1$ debe ser igual a 0:

$$\sum_{i=0}^{n-1} x_i * (i - 1) = 0$$

Donde x_i es el i-ésimo elemento de la secuencia mágica.

4. Acertijo Lógico

- Variables:
  - `apellido_juan, apellido_oscar, apellido_dario`
    Son las variables que corresponden a los apellidos de cada persona: Juan, Oscar y Dario. Los posibles valores que pueden tomar dichas variables son los elementos del dominio (tipo enumerado)

    `APELLIDOS`
  - `musica_juan, musica_oscar, musica_dario`
    Son las variables que corresponden a la musica preferida de cada persona: Juan, Oscar y Dario. Los posibles valores que pueden tomar dichas variables son los elementos del dominio (tipo enumerado)

    `MUSICA`
  - `edad_juan, edad_oscar, edad_dario`
    Son las variables que corresponden a la edad de cada persona: Juan, Oscar y Dario. Los posibles valores que pueden tomar dichas variables son los elementos del dominio (tipo enumerado)

    `EDADES`
- Dominios:
  - `APELLIDOS = {Gonzalez, Garcia, Lopez}`
    Los posibles valores de los apellidos, dados por el enunciado del problema.
  - `MUSICA = {Clasica, Pop, Jazz}`
    Los posibles valores de la música preferida, dados por el enunciado del problema.
  - `EDADES = 24..26`
    Los posibles valores de las edades, de 24 a 26 años.
- Restricciones: Antes podríamos considerar los siguientes predicados:
  1. $apellido(x, y)$ = La persona x se apellida y

2. $musica(x, y)$ = La música favorita de la persona x es y
3. $edad(x, y)$ = La persona x tiene y años

→ Restricciones:

- $\neg apellido(Juan, Gonzalez)$
  (Juan no se apellida González, esto se infiere al decir que Juan es mayor que González)
- $\neg apellido(Oscar, Lopez)$
  (Oscar no se apellida López)
- $\forall x \mid x \in APELLIDOS$
  $apellido(Juan, x_i) \wedge apellido(Oscar, x_{i+1}) \wedge apellido(Dario, x_{i+2}) \Rightarrow x_i \neq x_{i+1} \neq x_{i+2}$
  (Todos se deben apellidar distinto)
- $\neg musica(Dario, Jazz)$
  (A Dario no le gusta el Jazz)
- $\exists x \in APELLIDOS \wedge \exists y \in MUSICA \mid apellido(Oscar, x) \wedge x = Gonzalez \Rightarrow (musica(Oscar, y) \wedge y = Clasica)$
- $\exists x \in APELLIDOS \wedge \exists y \in MUSICA \mid apellido(Dario, x) \wedge x = Gonzalez \Rightarrow (musica(Dario, y) \wedge y = Clasica)$
  (Como Juan no puede ser González, solo Oscar y Dario pueden apellidarse González, y si alguno de los dos se apellida González, su música favorita debe ser la clásica)
- $\forall y \mid y \in MUSICA$
  $musica(Juan, y_i) \wedge musica(Oscar, y_{i+1}) \wedge musica(Dario, y_{i+2}) \Rightarrow y_i \neq y_{i+1} \neq y_{i+2}$
  (A cada uno le gusta un estilo musical distinto)
- $edad(Oscar, 25)$
  (Oscar tiene 25 años)
- $\exists x, y \in EDADES \mid$
  $apellido(Oscar, Gonzalez) \Rightarrow edad(Juan, x) \wedge edad(Oscar, y) \wedge x > y$
  (Si Oscar se apellida Gonzalez, como Juan debe ser mayor que Gonzalez, Juan debería ser mayor que Oscar)

- $\exists x, y \in EDADES \mid$
  $apellido(Dario, Gonzalez) \Rightarrow edad(Juan, x) \wedge edad(Dario, y) \wedge x > y$
  (Si Dario se apellida Gonzalez, como Juan debe ser mayor que Gonzalez, Juan debería ser mayor que Dario)
- $musica(Juan, Pop) \Rightarrow \exists x \in APELLIDOS \wedge \exists y \in EDADES \mid$
  $apellido(Juan, x) \wedge edad(Juan, y) \wedge x \neq Garcia \wedge y \neq 24$
  (Si Juan llegase a ser el fan de la música Pop, no puede apellidarse García ni tener 24 años)
- $musica(Oscar, Pop) \Rightarrow \exists x \in APELLIDOS \wedge \exists y \in EDADES \mid$
  $apellido(Oscar, x) \wedge edad(Oscar, y) \wedge x \neq Garcia \wedge y \neq 24$
  (Si Oscar llegase a ser el fan de la música Pop, no puede apellidarse García ni tener 24 años)
- $musica(Dario, Pop) \Rightarrow \exists x \in APELLIDOS \wedge \exists y \in EDADES \mid$
  $apellido(Dario, x) \wedge edad(Dario, y) \wedge x \neq Garcia \wedge y \neq 24$
  (Si Dario llegase a ser el fan de la música Pop, no puede apellidarse García ni tener 24 años)

5. Ubicación de personas en una reunión

- Variables:

  - `n`

    Denota el número n de personas en la reunión

  - `cantidad_nexts`

    Denota el número de restricciones de vecindad «next», es decir, cuantas duplas hay en el array de next

  - `cantidad_separates`

    Denota el número de restricciones de no vecindad «separate», es decir, cuantas duplas hay en el array de separate

  - `cantidad_distancias`

    Denota el número de restricciones de distancia entre personas «distancia», es decir, cuantas tripletas hay en el array de distancia

  - `entrada`

    Es el arreglo que contiene las personas que saldrán en la foto. Todos sus elementos deben pertenecer al dominio del conjunto enumerado

    `PERSONAS`

  - `next`

    Es el arreglo que contiene las restricciones de proximidad entre una persona en la foto y otra. (cada dupla significa que una persona debe estar al lado de la otra)

  - `separate`

    Es el arreglo que contiene las restricciones de NO proximidad entre una persona en la foto y otra. (cada dupla significa que una persona no puede estar al lado de la otra)

  - `distancia`

    Es el arreglo que contiene las restricciones sobre distancia de separación entre una persona u otra. (cada tripleta significa que la primera persona debe estar separada de la otra persona por máximo un determinado número de personas)

- `salida`

  Es el arreglo que contiene las variables de decisión sobre las cuales se aplican las restricciones apropiadas. Sus elementos también pertenecen al conjunto enumerado de

  `PERSONAS`

- Dominios:

  - `PERSONAS`

    El conjunto de las posibles personas que saldrán en la foto
  - `DOMINIO1 = 1..n`

    Dominio de 1 hasta el número n de personas en la foto
  - `DOMINIO2 = 1..cantidad_nexts`

    Dominio que es útil para saber cuántas restricciones de next hay
  - `DOMINIO3 = 1..cantidad_separates`

    Dominio que es útil para saber cuántas restricciones de separate hay
  - `DOMINIO4 = 1..cantidad_distancias`

    Dominio que es útil para saber cuántas restricciones de distancia hay

- Restricciones:

  - Todas las personas en la fila de la foto deben ser distintas:

    $$\forall x \mid x \in PERSONAS \land x_0, x_1, ..., x_{i-2}, x_{i-1} \in salida$$

    $$x_0 \neq x_1 \neq ... \neq x_{i-2} \neq x_{i-1}$$

  - Restricciones next:

    $$\forall i \in DOMINIO1, j \in DOMINIO2$$

    $$salida(i) = entrada(next(j,1)) \land i = 1 \Rightarrow salida(i+1) = entrada(next(j,2))$$

    $$salida(i) = entrada(next(j,1)) \land i = 8 \Rightarrow salida(i-1) = entrada(next(j,2))$$

    $$salida(i) = entrada(next(j,1)) \land (i \neq 1 \land i \neq 8) \Rightarrow$$

    $$salida(i+1) = entrada(next(j,2)) \lor salida(i-1) = entrada(next(j,2))$$

- Restricciones separate:

$$\forall i \in DOMINIO1, j \in DOMINIO3$$

$$salida(i) = entrada(separate(j,1)) \wedge i = 1 \Rightarrow salida(i{+}1) \neq entrada(separate(j,2))$$

$$salida(i) = entrada(separate(j,1)) \wedge i = 8 \Rightarrow salida(i{-}1) \neq entrada(separate(j,2))$$

$$salida(i) = entrada(separate(j,1)) \wedge (i \neq 1 \wedge i \neq 8) \Rightarrow$$

$$salida(i{+}1) \neq entrada(separate(j,2)) \vee salida(i{-}1) \neq entrada(separate(j,2))$$

- Restricciones distancia:

$$\forall i, k \in DOMINIO1, j \in DOMINIO4$$

$$salida(i) = entrada(distancia(j,1)) \wedge salida(k) = entrada(distancia(j,2)) \Rightarrow$$

$$|i - k| \leq distancia(j,3) + 1$$

PARTE 2 (COPs)

7. Construcción de Transformadores

- Variables:
  - a
- Dominios:
  - b
- Restricciones:
  - c
- Función objetivo:

8. Producción de Gasolina

- Variables:
  - a
- Dominios:
  - b
- Restricciones:
  - c
- Función objetivo: