LAB

Algoritmos y Programación

Semana 13: Programación con Restricciones

Dieta

RECORDATORIO
EXPLICADO EN CLASE DE TEORÏA

Table constraint

The `table` constraint enforces that a tuple of variables takes a value from a set of tuples. Since there are no tuples in MiniZinc this is encoded using arrays.

The usage of `table` has one of the forms

```
table(array[int] of var bool: x, array[int, int] of bool: t)
table(array[int] of var int:  x, array[int, int] of int:  t)
```

https://www.minizinc.org/doc-2.6.4/en/predicates.html#table

Ejemplo: Dieta equilibrada

RECORDATORIO
EXPLICADO EN CLASE DE TEORÏA

parámetros

Necesitamos encontrar una dieta completa de coste mínimo:
- calorías mínimas
- proteinas mínimas
- máximo de sal
- máximo de grasas

```
include "table.mzn";

int: min_energy;
int: min_protein;
int: max_salt;
int: max_fat;
set of FOOD: desserts;
set of FOOD: mains;
set of FOOD: sides;
enum FEATURE = { name, energy, protein, salt, fat, cost};
enum FOOD;
array[FOOD,FEATURE] of int: dd; % food database
```

```
set of int: FEATURES = 1..6;
int: name = 1; int: energy = 2; int: protein = 3;
int: salt = 4; int: fat = 5; int: cost = 6;
```

Declaraciones equivalentes de FEATURE utilizando set of int

Ejemplo: Dieta equilibrada

1) parámetros

```
include "table.mzn";

int: min_energy;
int: min_protein;
int: max_salt;
int: max_fat;
set of FOOD: desserts;
set of FOOD: mains;
set of FOOD: sides;
enum FEATURE = { name, energy, protein, salt, fat, cost};
enum FOOD;
array[FOOD,FEATURE] of int: dd; % food database
```

```
min_energy = 3300;
min_protein = 500;
max_salt = 180;
max_fat = 320;

FOOD     = { icecream, banana, chocolatecake, lasagna,
             steak, rice, chips, brocolli, beans} ;
desserts = { icecream, banana, chocolatecake };
mains    = { lasagna, steak, rice };
sides    = { chips, brocolli, beans };

dd = [| icecream,      1200,  50,  10, 120,  400     % icecream
      | banana,         800, 120,   5,  20,  120     % banana
      | chocolatecake, 2500, 400,  20, 100,  600     % chocolate cake
      | lasagna,       3000, 200, 100, 250,  450     % lasagna
      | steak,         1800, 800,  50, 100, 1200     % steak
      | rice,          1200,  50,   5,  20,  100     % rice
      | chips,         2000,  50, 200, 200,  250     % chips
      | brocolli,       700, 100,  10,  10,  125     % brocolli
      | beans,         1900, 250,  60,  90,  150 |]; % beans
```

name energy protein salt fat cost

RECORDATORIO
EXPLICADO EN CLASE DE TEORÏA

Ejemplo: Dieta equilibrada

RECORDATORIO
EXPLICADO EN CLASE DE TEORÏA

```
min_energy = 3300;
min_protein = 500;
max_salt = 180;
max_fat = 320;

FOOD     = { icecream, banana, chocolatecake, lasagna,
             steak, rice, chips, brocolli, beans} ;
desserts = { icecream, banana, chocolatecake };
mains    = { lasagna, steak, rice };
sides    = { chips, brocolli, beans };

dd = [| icecream,      1200,  50,  10, 120,  400     % icecream
      | banana,         800, 120,   5,  20,  120     % banana
      | chocolatecake, 2500, 400,  20, 100,  600     % chocolate cake
      | lasagna,       3000, 200, 100, 250,  450     % lasagna
      | steak,         1800, 800,  50, 100, 1200     % steak
      | rice,          1200,  50,   5,  20,  100     % rice
      | chips,         2000,  50, 200, 200,  250     % chips
      | brocolli,       700, 100,  10,  10,  125     % brocolli
      | beans,         1900, 250,  60,  90,  150 |]; % beans
```

name energy protein salt fat cost

2) variables de decisión

```
array[FEATURE] of var int: main;
array[FEATURE] of var int: side;
array[FEATURE] of var int: dessert;
var int: budget;
```

Ejemplo: Dieta equilibrada

3) restricciones

RECORDATORIO
EXPLICADO EN CLASE DE TEORÏA

```
min_energy = 3300;
min_protein = 500;
max_salt = 180;
max_fat = 320;

FOOD     = { icecream, banana, chocolatecake, lasagna,
             steak, rice, chips, brocolli, beans} ;
desserts = { icecream, banana, chocolatecake };
mains    = { lasagna, steak, rice };
sides    = { chips, brocolli, beans };

dd = [| icecream,      1200,  50,  10, 120,  400     % icecream
      | banana,         800, 120,   5,  20,  120     % banana
      | chocolatecake, 2500, 400,  20, 100,  600     % chocolate cake
      | lasagna,       3000, 200, 100, 250,  450     % lasagna
      | steak,         1800, 800,  50, 100, 1200     % steak
      | rice,          1200,  50,   5,  20,  100     % rice
      | chips,         2000,  50, 200, 200,  250     % chips
      | brocolli,       700, 100,  10,  10,  125     % brocolli
      | beans,         1900, 250,  60,  90,  150 |]; % beans
```

name energy protein salt fat cost

```
constraint main[name] in mains;
constraint side[name] in sides;
constraint dessert[name] in desserts;

constraint table(main, dd);
constraint table(side, dd);
constraint table(dessert, dd);

constraint main[energy] + side[energy] + dessert[energy] >=min_energy;
constraint main[protein]+side[protein]+dessert[protein] >=min_protein;
constraint main[salt] + side[salt] + dessert[salt] <= max_salt;
constraint main[fat] + side[fat] + dessert[fat] <= max_fat;

constraint budget = main[cost] + side[cost] + dessert[cost];
solve minimize budget;
```

Ejemplo: Dieta equilibrada

RECORDATORIO
EXPLICADO EN CLASE DE TEORÏA

```
min_energy = 3300;
min_protein = 500;
max_salt = 180;
max_fat = 320;

FOOD     = { icecream, banana, chocolatecake, lasagna,
             steak, rice, chips, brocolli, beans} ;
desserts = { icecream, banana, chocolatecake };
mains    = { lasagna, steak, rice };
sides    = { chips, brocolli, beans };

dd = [| icecream,      1200,  50,  10, 120,  400     % icecream
      | banana,         800, 120,   5,  20,  120     % banana
      | chocolatecake, 2500, 400,  20, 100,  600     % chocolate cake
      | lasagna,       3000, 200, 100, 250,  450     % lasagna
      | steak,         1800, 800,  50, 100, 1200     % steak
      | rice,          1200,  50,   5,  20,  100     % rice
      | chips,         2000,  50, 200, 200,  250     % chips
      | brocolli,       700, 100,  10,  10,  125     % brocolli
      | beans,         1900, 250,  60,  90,  150 |]; % beans
```

name energy protein salt fat cost

```
Running meal.mzn, meal.dzn
main = steak, side = brocolli, dessert = banana, cost = 1445
----------
main = rice, side = brocolli, dessert = chocolatecake, cost = 825
----------
==========
Finished in 79msec.
```

solución incumbente

solución óptima

VPL 1: Dieta con restricción de comidas especiales

- Modifica el modelo de la dieta equilibrada del tutorial de MiniZinc para que incorpore los siguientes cambios:
  - Parámetros adicionales:
    - Algunas comidas son consideradas especiales
    - Máximo de comidas especiales

Parámetros adicionales

```
set of FOOD: special_foods;
int: max_special_foods;
```

- Restricción adicional:
  - Nuestra dieta equilibrada sólo puede contener max_especial_foods comidas especiales

De las comidas que tenemos en el conjunto 'special_foods' nuestra dieta sólo puede contener el máximo de comidas que nos indica el parámetro 'max_special_foods'

Ejemplo

```
special_foods = {icecream, chocolatecake, spaghetti, mashedpotato, banana};
max_special_foods = 2;

output=main = spaghetti, side = beans, dessert = banana, cost = 720
----------
==========
```

VPL 1: Dieta con restricción de comidas especiales

dieta.mzn

```
include "table.mzn";
int: min_energy;
int: min_protein;
int: max_salt;
int: max_fat;

set of FOOD: desserts;
set of FOOD: mains;
set of FOOD: sides;
enum FEATURE = { name, energy, protein, salt, fat, cost};
enum FOOD;
array[FOOD, FEATURE] of int: dd;

set of FOOD: special_foods;   % Conjunto de comidas especiales
int: max_special_foods;       % Máximo de comidas especiales que podemos utilizar

array[FEATURE] of var int: main;
array[FEATURE] of var int: side;
array[FEATURE] of var int: dessert;
var int: budget;

output ["main = ", show(to_enum(FOOD,main[name])),
        ", side = ", show(to_enum(FOOD,side[name])),
        ", dessert = ", show(to_enum(FOOD,dessert[name])),
        ", cost = ", show(budget), "\n"];

% Escribe el código a partir de aquí -------------------------------
```

VPL 2: Dieta con restricción de comidas especiales y varios platos principales

- Modifica tu solución del ejercicio anterior para que elija varios platos principales, 1 acompañamiento y 1 postre.
  - Parámetro adicional
    - Número de platos principales (num_main)

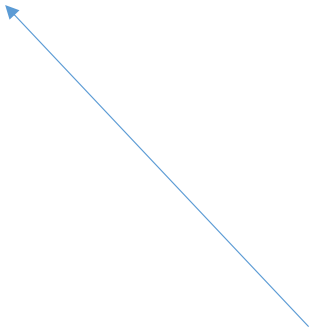
```
set of FOOD: special_foods;
int: max_special_foods;
int: num_main;                                    % Número de platos principales

array[FEATURE, 1..num_main] of var int: main;  % Ahora es una matriz
array[FEATURE] of var int: side;
array[FEATURE] of var int: dessert;
var 0..infinity: budget;
```

Estas son las diferencias de este VPL con respecto al anterior

Ejemplo

```
special_foods = {icecream, chocolatecake, spaghetti,
                 mashedpotato, banana, rice, croquette};
max_special_foods = 1;
num_main = 2;
output=main= soup, rice side = puree, dessert = strawberry, cost = 825
```



Ahora se eligen 2 platos principales (2 main)

MiniZinc: Sintaxis para seleccionar filas o columnas completas de la matriz

- La sintaxis '..' en uno de los índices de acceso a una matriz nos proporciona un vector que contiene todos los elementos de una determinada fila o columna de una matriz.

Ejemplo:

```
include "globals.mzn";

int: n = 2;
array [1..n, 1..n] of var 1..n : m;

% Todos los elementos de cada fila diferentes
constraint forall(i in 1..n)(alldifferent(m[i, ..]));

% Todos los elementos de cada columna diferentes
constraint forall(j in 1..n)(alldifferent(m[.., j]));

solve satisfy;
```

vector con todos los elementos de la fila i

vector con todos los elementos de la columna j

```
Running 04.09-rows-cols-syntax.mzn
m =
[| 1, 2
 | 2, 1
 |];
----------
m =
[| 2, 1
 | 1, 2
 |];
----------
==========
Finished in 114msec.
```