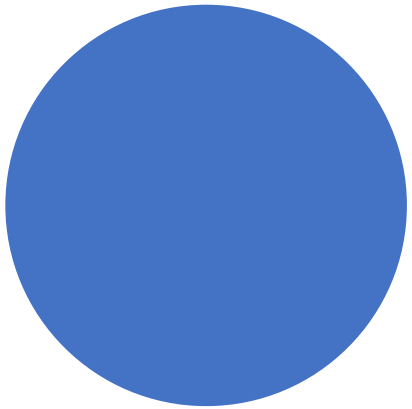
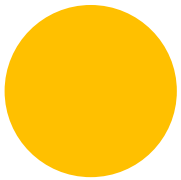



Algoritmos y Programación

Práctica 9a
Programación con Restricciones

Minizinc

- Estructura básica
  1. **include** ⟨*filename*⟩;
  2. Declaración de variables
     1. **Parámetros**: float/range of float, int/range of int, string, bool, ann
     2. **Variables de decisión**: float/range of float, int/range of int, bool
     3. ⟨*type inst expr*⟩: ⟨*variable*⟩ [= ⟨*expression*⟩];
  3. Asignación de valores a variables: ⟨*variable*⟩ = ⟨*expression*⟩;
  4. Restricciones: **constraint** ⟨*Boolean expression*⟩;
  5. solve satisfy;

     solve maximize ⟨*arithmetic expression*⟩;

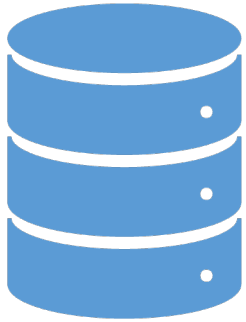
     solve minimize ⟨*arithmetic expression*⟩;
  6. **output** [⟨*string expression*⟩, ▪ ▪ ▪ , ⟨*string expression*⟩];

Minizinc

- Hello World – **hello.mzn**
  - Construir un modelo que muestre en pantalla “Hello World”
  - Ejecutar desde el IDE de Minizinc.
  - [Opcional] Ejecutar desde consola.

Minizinc:

- **output**
- **show**

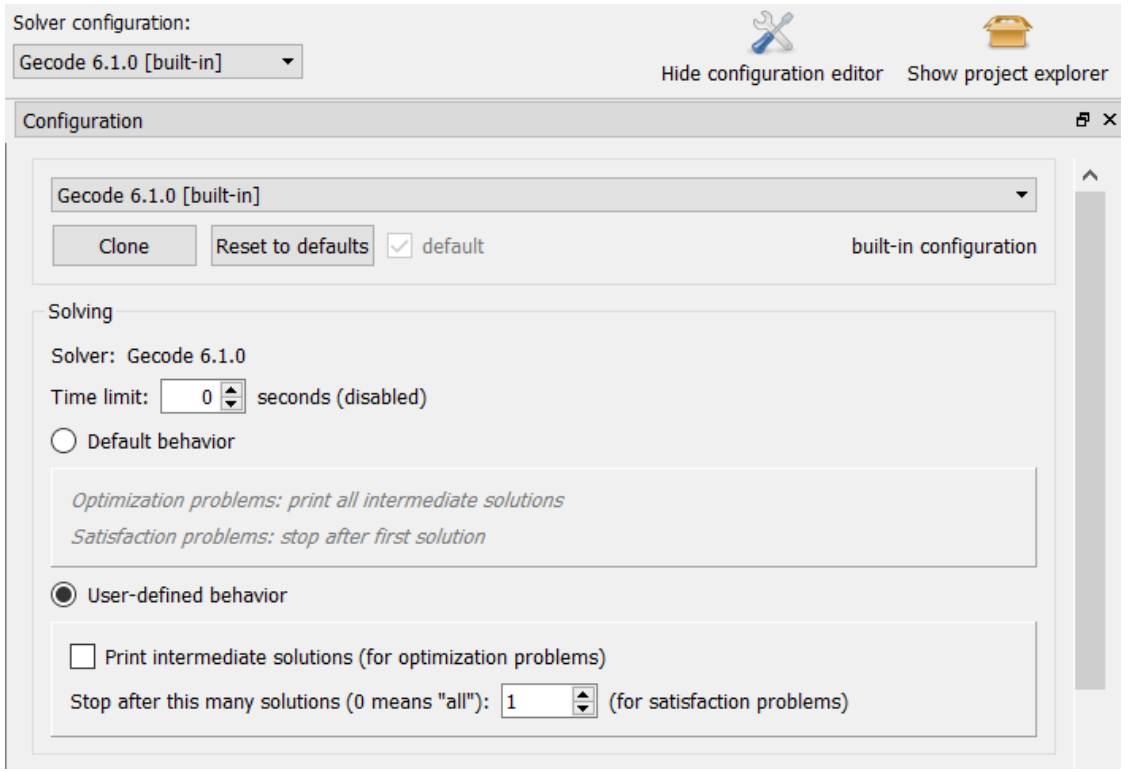


Minizinc

- Input and Output – **inout.mzn**
  - Construir un modelo inout.mzn que defina un parámetro entero y lo muestre en pantalla
    - Desde el IDE debería saltar un pop up
    - [Opcional] Desde consola: minizinc inout.mzn –D”n = 23;”

Minizinc

- **Variable de decisión**
  - Construir un modelo **xvar.mzn** con una variable de decisión en el rango 1..10 y que muestre todos sus valores
    - Desde el IDE:
      - Tab de configuración
      - User-defined behavior
      - “Stop after this many solutions (0 means “all”)
    - [Opcional] Desde consola: minizinc xvar.mzn -a



Minizinc

- Optimización
  - Construir un modelo **xoptima.mzn** con una variable de decision, x, en el rango 0..10 con la restricción de que x debe ser divisible por 4, la salida debe ser aquella que minimice el valor $(x-7)^2$
    - Desde el IDE:
      - Tab de configuración
      - User-defined behavior
      - “Stop after this many solutions (0 means “all”)
    - [Opcional] Desde consola:
      - minizinc xoptima.mzn –a, devuelve todas las soluciones
      - miniznc xoptima.mzn: devuelve el óptimo

Minizinc:

- **mod**
- **solve minimize …**

Minizinc

- Arrays
  - Construir un modelo **array.mzn**:
    - Parámetro n, que define la longitud del array
    - El array contiene variables de decisión con valores entre 0..9
    - Restricción, la suma de los valores del array es igual al producto de los valores del array
    - La salida debe ser el array resultante

  - Añadir la restricción de que el orden de los valores del array es creciente
    - $x[1] \leq x[2] \ldots \leq x[n]$

Para pasar bien este ejemplo en el VPL, en vez de 'solve satisfy' deberás maximizar el producto de los valores del array.

Minizinc:

- **sum**
- **product**
- **forall**
- **solve satisfy**