

# Advanced School in Artificial Intelligence

## Il linguaggio MiniZinc

Marco Gavanelli

[marco.gavanelli@unife.it](mailto:marco.gavanelli@unife.it)




*Progetto di alta formazione in ambito tecnologico economico e culturale per una regione della conoscenza europea e attrattiva approvato e cofinanziato dalla Regione Emilia-Romagna con deliberazione di Giunta regionale n. 1625/2021*



**Università  
degli Studi  
di Ferrara**

## Il linguaggio MiniZinc

- È un linguaggio per descrivere problemi di soddisfacimento di vincoli (e problemi di ottimizzazione)
- Permette quindi di definire: 
  - Variabili
  - Domini
  - Vincoli
  - Funzione obiettivo



## MiniZinc e FlatZinc

- Le specifiche scritte in MiniZinc vengono convertite in linguaggio **FlatZinc** tramite un convertitore mzn2fzn
- Le specifiche FlatZinc possono poi essere risolte tramite molti solver
  - COIN-OR CBC
  - Gurobi
  - IBM ILOG CPLEX
  - Gecode
  - Chuffed
  - Choco 3
  - ECLiPSe
  - HaifaCSP
  - JaCoP
  - MinisatID
  - Mistral 2.0
  - Opturion CPX
  - OR-Tools
  - OscaR/CBLS
  - Picat
  - SICStus
  - SCIP
  - Yuck

Più alcuni solver sviluppati per MiniZinc, già integrati in un IDE


Questi solver sono basati su diverse tecnologie; molti sono basati su Constraint Programming, per cui ci focalizzeremo su questa tipologia di solver. Altri solver sono basati su MILP o su SAT

## MiniZinc: Variabili e domini

- Per definire una variabile intera e il suo dominio

```
var 1..10 : nome; 
```

- In questo modo dichiariamo che esiste una variabile chiamata **nome** il cui dominio va da 1 a 10.
- Oppure

```
var {1,7,8,10} : nome; 
```

- Oppure:

```
var int : nome;
```

- se non si vuole definire il dominio  
(Dominio a default –MaxInt...Maxint)



## MiniZinc: Vincoli

- Per definire un vincolo, si usa la parola chiave **constraint**
- Ad esempio

```
var 1..10 : x;  
var 0..5  : y;  
constraint x<=y;
```

## MiniZinc: Far partire la ricerca

- Per far partire la ricerca di una soluzione, si usa

```
solve satisfy;
```

- Ad esempio

```
var 1..10 : x;  
var 0..5 : y;  
constraint x<=y;  
solve satisfy;
```

```
Compiling prova.mzn  
Running prova.mzn  
x = 1;  
y = 1;  
-----  
Finished in 9msec
```

- cerca una qualunque soluzione che soddisfi i vincoli

## MiniZinc: Altri Vincoli

- Alcuni vincoli:
  - $x > y$   $x$  è maggiore di  $y$
  - $x < y$   $x$  è minore di  $y$
  - $x = y$   $x$  è uguale a  $y$
  - $x \neq y$   $x$  è diverso da  $y$
  - $x \geq y$   $x$  è maggiore o uguale a  $y$
  - $x \leq y$   $x$  è minore o uguale a  $y$
- All'interno dei vincoli si possono usare gli operatori:
  - $+$  somma
  - $-$  sottrazione
  - $*$  prodotto
  - $\text{div}$  divisione intera
  - $\text{mod}$  resto della divisione intera
  - $\text{abs}$  valore assoluto
  - $\text{pow}(b,e)$  potenza  $b^e$

Ad esempio:

**constraint**  $x*y \leq x+2*z-k*k/z;$

## Esercizio: scrivere in MiniZinc l'esercizio precedente

- Tre bimbi giocano a biglie; ciascuno dei 3 ha un numero di biglie da 1 a 5.
- Aldo è quello che ne ha di più.
- Bruno e Carlo, insieme, ne hanno 5.
- Bruno ha un numero di biglie diverso da quello che ha Carlo