

MiniZinc

# MiniZinc Syntax

MiniZinc allows the user to specify:

- **Parameters**
  - MiniZinc also enables the **separation between model and data**.
- **Variables** of different **type** and corresponding **domains**
  - Boolean, Integers, Floats, Set of integers
    - `var bool: B; var -2..10: I; var 0.0..1.0: F; var set of 1..10: S;`
- **Constraints** over such variables
  - Arithmetical, logical, **global**, ...
    - E.g. `y = 2*x + z`  $\vee$  `b != 4; all_different([x,y,z]);`
- Both **CSPs** and **COPs** (minimization/maximization)

# Example


- Subset-sum problem: are there N numbers in a set S adding up to K?

```
1 % Are there N numbers in a set S adding up to K?  
2 include "globals.mzn";  
3  
4 set of int: S = {7, 10, 23, 13, 4, 16};  
5 int: N = 4;  
6 int: K = 50;  
7  
8 array[1..N] of var S: X;  
9 constraint all_different(X);  
10 constraint sum(X) = K;  
11  
12 solve satisfy;
```

# Example

- Subset-sum problem: are there N numbers in a set S adding up to K?

```
1 % Are there N numbers in a set S adding up to K?  
2 include "globals.mzn";  
3  
4 set of int: S = {7, 10, 23, 13, 4, 16};  
5 int: N = 4;  
6 int: K = 50;  
7  
8 array[1..N] of var S: X;  
9 constraint all_different(X);  
10 constraint sum(X) = K;  
11  
12 solve satisfy;
```



For using global constraints

# Example

- Subset-sum problem: are there N numbers in a set S adding up to K?

Parameters

```
1 % Are there N numbers in a set S adding up to K?  
2 include "globals.mzn";  
3  
4 set of int: S = {7, 10, 23, 13, 4, 16};  
5 int: N = 4;  
6 int: K = 50;  
7  
8 array[1..N] of var S: X;  
9 constraint all_different(X);  
10 constraint sum(X) = K;  
11  
12 solve satisfy;
```

For using global constraints

# Example

- Subset-sum problem: are there N numbers in a set S adding up to K?

Parameters

```
1 % Are there N numbers in a set S adding up to K?
2 include "globals.mzn";
3
4 set of int: S = {7, 10, 23, 13, 4, 16};
5 int: N = 4;
6 int: K = 50;
7
8 array[1..N] of var S: X;
9 constraint all_different(X);
10 constraint sum(X) = K;
11
12 solve satisfy;
```

For using global constraints

Instead of using N integer variables  $X_1, \dots, X_N$  we use an array X of N integer variables.

# Example

- Subset-sum problem: are there N numbers in a set S adding up to K?

Parameters

```
1 % Are there N numbers in a set S adding up to K?
2 include "globals.mzn";
3
4 set of int: S = {7, 10, 23, 13, 4, 16};
5 int: N = 4;
6 int: K = 50;
7
8 array[1..N] of var S: X;
9 constraint all_different(X);
10 constraint sum(X) = K;
11
12 solve satisfy;
```

For using global constraints

Instead of using N integer variables  $X_1, \dots, X_N$  we use an array X of N integer variables.

Global constraints: defined on an arbitrary number of variables.

# Example

- Subset-sum problem: are there N numbers in a set S adding up to K?

Parameters

```
1 % Are there N numbers in a set S adding up to K?
2 include "globals.mzn";
3
4 set of int: S = {7, 10, 23, 13, 4, 16};
5 int: N = 4;
6 int: K = 50;
7
8 array[1..N] of var S: X;
9 constraint all_different(X);
10 constraint sum(X) = K;
11
12 solve satisfy;
```

For using global constraints

Instead of using N integer variables  $X_1, \dots, X_N$  we use an array X of N integer variables.

Global constraints: defined on an arbitrary number of variables.

CSP



# Exercise 1

$$\begin{array}{rccccccccc} & & S & & E & & N & & D \\ + & & M & & O & & R & & E \\ \hline M & & O & & N & & E & & Y \end{array}$$

- Each letter is a distinct digit.
- $M \neq 0$ .
- $S \neq 0$ .

# Exercise 1

Step 1: Definition of the variables and their domains

# Exercise 1

## Step 1: Definition of the variables and their domains

```
% Parameters that define range of integers
set of int: DIGIT = 0..9;
set of int: DIGIT1 = 1..9;


% One variable for each letter of the problem
var DIGIT1: S;
var DIGIT: E;
var DIGIT: N;
var DIGIT: D;
var DIGIT1: M;
var DIGIT: O;
var DIGIT: R;
var DIGIT: Y;
```

# Exercise 1

## Step 1: Definition of the variables and their domains

```
% Parameters that define range of integers
set of int: DIGIT = 0..9;
set of int: DIGIT1 = 1..9;

% One variable for each letter of the problem
var DIGIT1: S;
var DIGIT: E;
var DIGIT: N;
var DIGIT: D;
var DIGIT1: M;
var DIGIT: O;
var DIGIT: R;
var DIGIT: Y;
```



This syntax allows us to define ranges, that help us defining variables' domains.

# Exercise 1

Step 2: Constraint definition

# Exercise 1

## Step 2: Constraint definition

- Sum of the variables

```
% Constraint for the sum SEND+MORE=MONEY
constraint 1000*S + 100*E + 10*N + D
           + 1000*M + 100*O + 10*R + E
=10000*M + 1000*O + 100*N + 10*E + Y;
```

# Exercise 1

## Step 2: Constraint definition

- All the variables have to be different

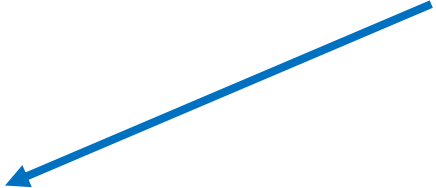
```
% Array of letters  
array[1..8] of var int: letters = [S,E,N,D,M,O,R,Y];  
  
% Constraint to ensure all letters are different  
constraint all_different(letters);
```

# Exercise 1

## Step 2: Constraint definition

- All the variables have to be different

Creating an array of variables is helpful when we have to create constraints on all of them.



```
% Array of letters  
array[1..8] of var int: letters = [S,E,N,D,M,O,R,Y];  
  
% Constraint to ensure all letters are different  
constraint all_different(letters);
```




# Exercise 1

## Step 2: Constraint definition

- All the variables have to be different

```
% Array of letters  
array[1..8] of var int: letters = [S,E,N,D,M,O,R,Y];  
  
% Constraint to ensure all letters are different  
constraint all_different(letters);
```



all\_different is a built-in constraint, we need to include it using the correct syntax.

# Exercise 1

Complete solution.

Note the inclusion of the `all_different` constraint and the “`solve satisfy;`” line, that tells MiniZinc to try to solve the problem.

```
1 % Send more money in minizinc.
2
3 include "all_different.mzn";
4
5 % Parameters that define range of integers
6 set of int: DIGIT = 0..9;
7 set of int: DIGIT1 = 1..9;
8
9 % One variable for each letter of the problem
10 var DIGIT1: S;
11 var DIGIT: E;
12 var DIGIT: N;
13 var DIGIT: D;
14 var DIGIT1: M;
15 var DIGIT: O;
16 var DIGIT: R;
17 var DIGIT: Y;
18
19 % Array of letters
20 array[1..8] of var int: letters = [S,E,N,D,M,O,R,Y];
21
22 % Constraint for the sum SEND+MORE=MONEY
23 constraint 1000*S + 100*E + 10*N + D
24           + 1000*M + 100*O + 10*R + E
25 = 10000*M + 1000*O + 100*N + 10*E + Y;
26
27 % Constraint to ensure all letters are different
28 constraint all_different(letters);
29
30 solve satisfy;
```

## Exercise 2 (February 11, 2021)

Anton, Beth, Carlos, and Daniel must split the restaurant bill of 80 euros. The place does not accept credit cards, so they have to pay using paper bills (banknotes) and coins.

- Anton has only two 20 euros bills, so he can pay only 20 or 40.
- Carlos has a debt with Beth, so he will pay 13 euros more than her.
- Daniel is very prideful, so he will not pay less than Carlos.
- None of them wants to pay less than 10 euros nor more than half of the bill (40 euros).

Write a CLP or MiniZinc program to compute how they can split the bill. Please use comments so to make clear what is your reasoning and which variables will contain the final results.

# Exercise 2

## Step 1: Definition of the variables and their domains

```
% Parameters that define range of integers  
set of int: MONEY = 10..40;
```

```
% One variable for each of the 4 friends  
var MONEY: A;  
var MONEY: B;  
var MONEY: C;  
var MONEY: D;
```

# Exercise 2

## Step 2: Constraint definition

- Anton has only two 20 euros bills, so he can pay only 20 or 40.

```
constraint A = 20 \ / A = 40;
```

# Exercise 2

## Step 2: Constraint definition

- Carlos has a debt with Beth, so he will pay 13 euros more than her.

```
constraint C = 13 + B;
```

# Exercise 2

## Step 2: Constraint definition

- Daniel is very prideful, so he will not pay less than Carlos.

```
constraint D > C;
```

# Exercise 2

## Step 2: Constraint definition

- None of them wants to pay less than 10 euros nor more than half of the bill (40 euros).

```
% Creating an array of variables  
array[1..4] of var int: people = [A,B,C,D];  
% Creating a single constraint instead of 8  
constraint forall(p in people) (p >= 10 /\ p <= 40);
```



# Exercise 2

## Step 2: Constraint definition

- The bill is 80 euros.

```
constraint A + B + C + D = 80;
```

# Exercise 2

Complete  
solution.

```
1 % Bill split among 4 firends (LAAI exam 2021.02.11)
2
3 % Parameters that define range of integers
4 set of int: MONEY = 10..40;
5
6 % One variable for each of the 4 friends
7 var MONEY: A;
8 var MONEY: B;
9 var MONEY: C;
10 var MONEY: D;
11
12 % Anton has only two 20 euros bills, so he can pay only 20 or 40
13 constraint A = 20 \/ A = 40;
14
15 % Carlos has a debt with Beth, so he will pay 13 euros more than her.
16 constraint C = 13 + B;
17
18 % Daniel is very prideful, so he will not pay less than Carlos.
19 constraint D > C;
20
21 % None of them wants to pay less than 10 euros nor more than half of the bill (40 euros).
22 % Creating an array of variables
23 array[1..4] of var int: people = [A,B,C,D];
24 % Creating a single constraint instead of 8
25 constraint forall(p in people) (p >= 10 /\ p <= 40);
26
27 % The total has to be 80 euros
28 constraint A + B + C + D = 80;
29
30 solve satisfy;
```

## Exercise 3 (January 18, 2021)

Morgan is a traveling salesperson that must visit three clients: Anton, Beth, and Carlos.

The house of Morgan is distant 3 hours from each client. The house of Anton is distant 2 hours from Beth, and 3 from Carlos. The houses of Beth and Carlos are only 1 hour distant from each other. Beth must be visited before 14, because she is very busy in the afternoon.

Knowing that Morgan won't leave the house before 9 and want to be back at home at most at 18, write a CLP program or a MiniZinc to compute the possible timetables for Morgan, including leaving and returning home.

# Exercise 3

## Step 1: Definition of the variables and their domains

```
% The earliest that Morgan can leave his house is at 9
% The latest that Morgan goes back home is 18
set of int: TIME = 9..18;
set of int: TIMEB = 9..14;

var TIME: H1;
var TIME: A;
var TIMEB: B;
var TIME: C;
var TIME: H2;
```

# Exercise 3

## Step 2: Constraint definition

- Morgan cannot be in two different places at the same time.

```
array[1..5] of var int: variables = [H1, A, B, C, H2];  
  
% Morgan will be in different places at different times  
constraint all_different(variables);
```

# Exercise 3

## Step 2: Constraint definition

- Morgan's travel starts from his house.

```
% Morgan starts from his house  
constraint H1 < A;  
constraint H1 < B;  
constraint H1 < C;
```

# Exercise 3

## Step 2: Constraint definition

- Morgan's travel ends at his house.

```
% Morgan ends up at his house  
constraint H2 > H1;  
constraint H2 > A;  
constraint H2 > B;  
constraint H2 > C;
```

# Exercise 3

## Step 2: Constraint definition

- Distances between the different houses.

```
% Distances between the different houses
constraint abs(H1 - A) >= 3;
constraint abs(H1 - B) >= 3;
constraint abs(H1 - C) >= 3;
constraint abs(H2 - A) >= 3;
constraint abs(H2 - B) >= 3;
constraint abs(H2 - C) >= 3;
constraint abs(A - B) >= 2;
constraint abs(A - C) >= 3;
constraint abs(B - C) >= 1;
```



# Exercise 3

## Complete solution.

```
1 % Salesman
2 include "all_different.mzn";
3
4 % The earliest that Morgan can leave his house is at 9
5 % The latest that Morgan goes back home is 18
6 set of int: TIME = 9..18;
7 set of int: TIMEB = 9..14;
8
9 var TIME: H1;
10 var TIME: A;
11 var TIMEB: B;
12 var TIME: C;
13 var TIME: H2;
14
15 array[1..5] of var int: variables = [H1, A, B, C, H2];
16
17 % Morgan will be in different places at different times
18 constraint all_different(variables);
19
20 % Morgan starts from his house
21 constraint H1 < A;
22 constraint H1 < B;
23 constraint H1 < C;
24
```

```
25 % Morgan ends up at his house
26 constraint H2 > H1;
27 constraint H2 > A;
28 constraint H2 > B;
29 constraint H2 > C;
30
31 % Distances between the different houses
32 constraint abs(H1 - A) >= 3;
33 constraint abs(H1 - B) >= 3;
34 constraint abs(H1 - C) >= 3;
35 constraint abs(H2 - A) >= 3;
36 constraint abs(H2 - B) >= 3;
37 constraint abs(H2 - C) >= 3;
38 constraint abs(A - B) >= 2;
39 constraint abs(A - C) >= 3;
40 constraint abs(B - C) >= 1;
41
42 solve satisfy;
```

# Prolog

Derivation

1) Show the derivation of the goal **p(a, b, W)** in the following Prolog program

`p (X ,Y , Z ):- q (X ,Y , Z ) , q (X ,X , Z ) .`

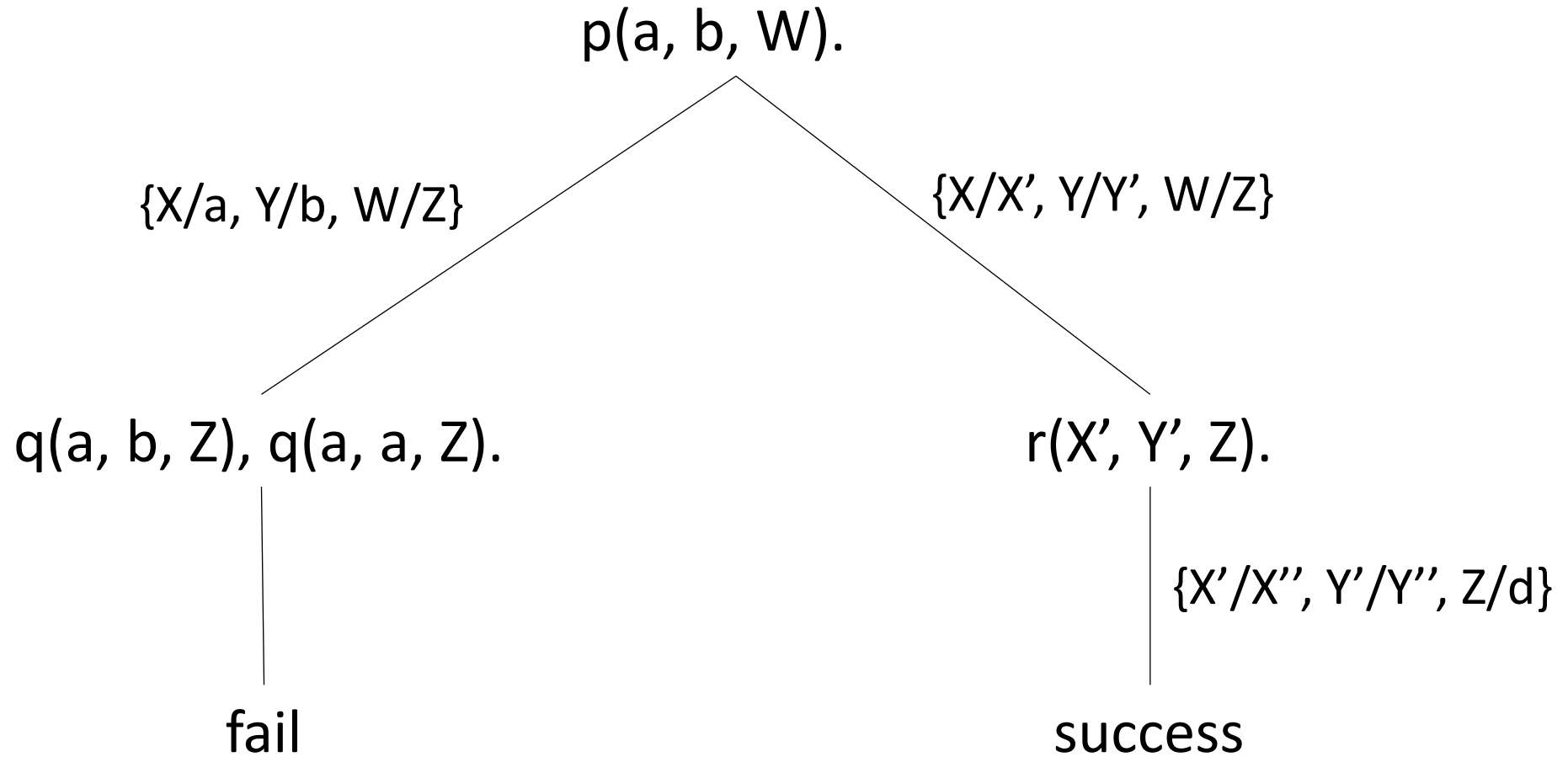
`p (X ,Y , Z ):- r (X ,Y , Z ) .`

`q (X ,X , c ) .`

`q (X ,a , c ) .`

`r (X ,Y , d ) .`

# 1) Solution



2) Show the derivation of the goal  $p(V, c, W)$  in the following Prolog program

$p(X, b, Z) :- q(X, Y, Z), q(X, X, Z).$

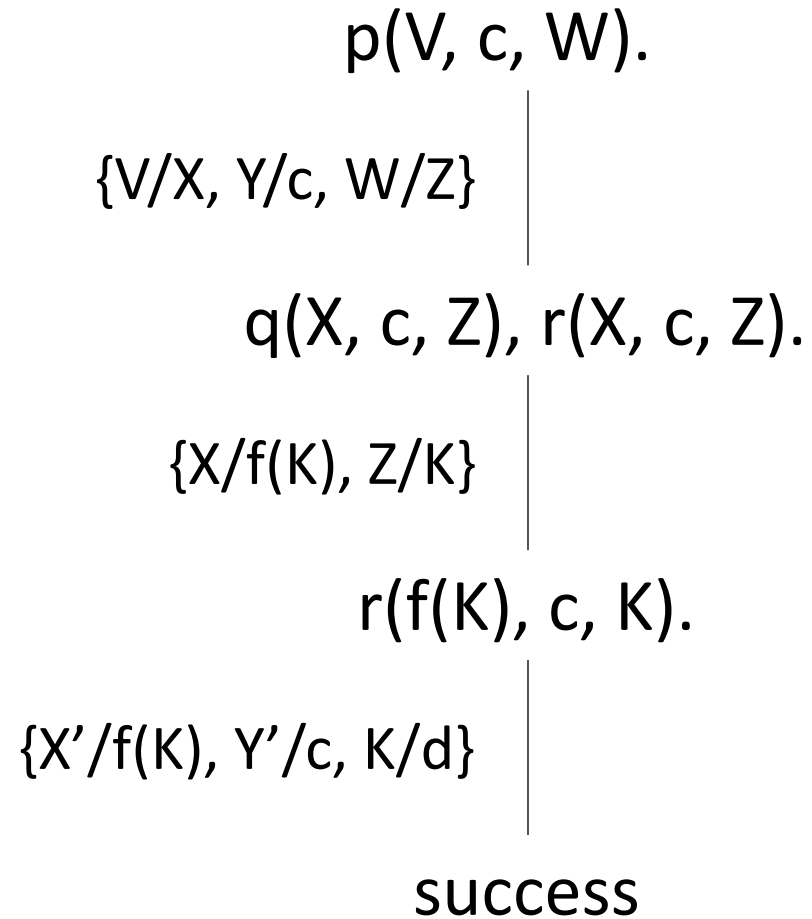
$p(X, Y, Z) :- q(X, Y, Z), r(X, Y, Z).$

$q(X, b, c).$

$q(f(K), c, K).$

$r(X, Y, d).$

## 2) Solution



3) Show the derivation of the goal  $p(W, Z)$  in the following Prolog program

$p(X, Y) :- \neg q(X, Y), q(Y, X).$

$q(f(V), g(V)) :- \neg r(V).$

$q(f(V), f(V)) :- \neg s(V).$

$r(a).$

$s(b).$

### 3) Solution

