Oxiflex - A Constraint Programming Solver for MiniZinc written in Rust

Gianluca Klimmer

University of Basel

15.07.2024

Constraint Programming Solver for MiniZinc written in Rust

Constraint Programming Solver for MiniZinc written in Rust

Constraint Network

A (binary) **constraint network** is a 3-tuple $C = \langle V, \text{dom}, (R_{uv}) \rangle$ such that:

- V is a non-empty and finite set of variables,
- dom is a function that assigns a non-empty and finite domain to each variable $v \in V$, and
- $(R_{uv})_{u,v \in V, u \neq v}$ is a family of binary relations (constraints) over V where for all $u \neq v : R_{uv} \subseteq \text{dom}(u) \times \text{dom}(v)$

Constraint Network

boring...

Constraint Network

- Variables
  - Values to choose from
- Constraints
  - Rules for choosing values

Simple Example

Variables:

$$w = \{1, 2, 3, 4\}$$
$$y = \{1, 2, 3, 4\}$$
$$x = \{1, 2, 3\}$$
$$z = \{1, 2, 3\}$$

Constraints:

$$w = 2 \cdot x$$
$$w < z$$
$$y > z$$

Constraint Programming

var 1..4: w;
var 1..4: y;
var 1..3: x;
var 1..3: z;

constraint $w = 2 \cdot x$;
constraint $w < z$;
constraint $y > z$;

Constraint Programming

```
var 1..4: w;
var 1..4: y;
var 1..3: x;
var 1..3: z;

constraint w = 2 · x;
constraint w < z;
constraint y > z;

solve satisfy;
```

Constraint Programming

var 1..4: w;
var 1..4: y;
var 1..3: x;
var 1..3: z;

constraint $w = 2 \cdot x$;
constraint $w < z$;
constraint $y > z$;

solve satisfy;

$\rightarrow$ MiniZinc!

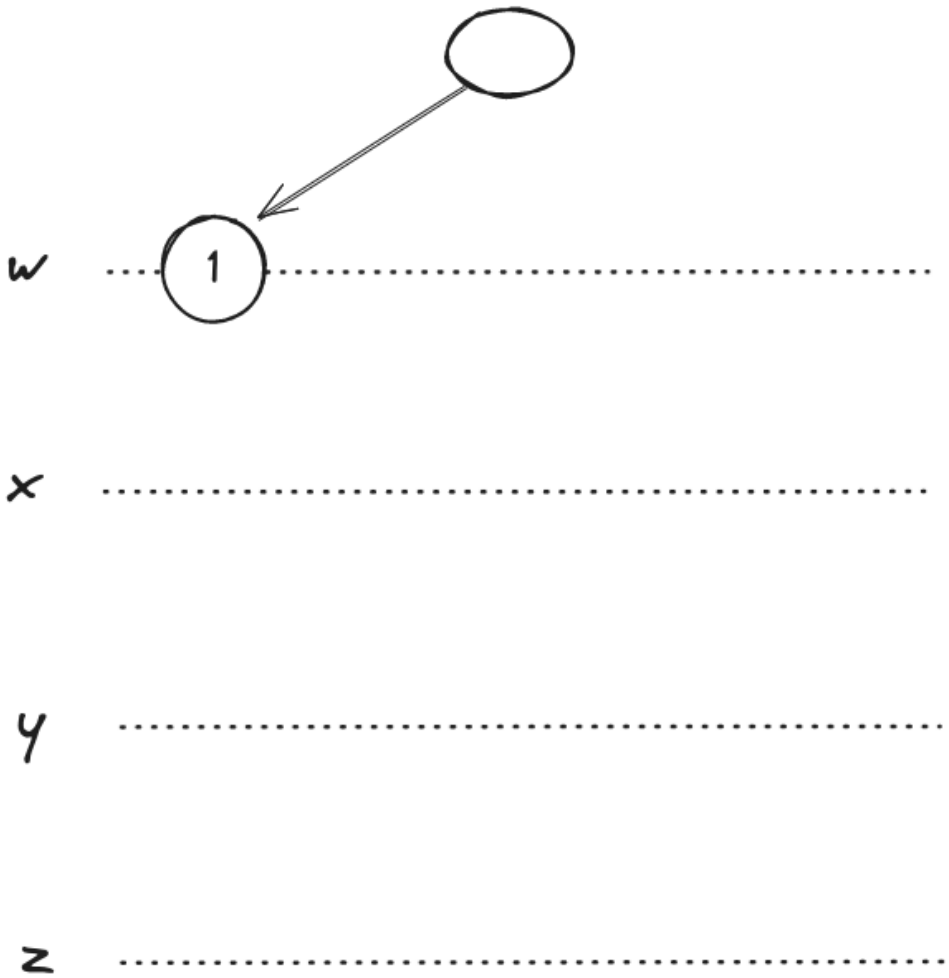
MiniZinc

Constraint Programming Solver for MiniZinc written in Rust

Constraint Programming Solver for MiniZinc written in Rust

Backtracking

Backtracking Example



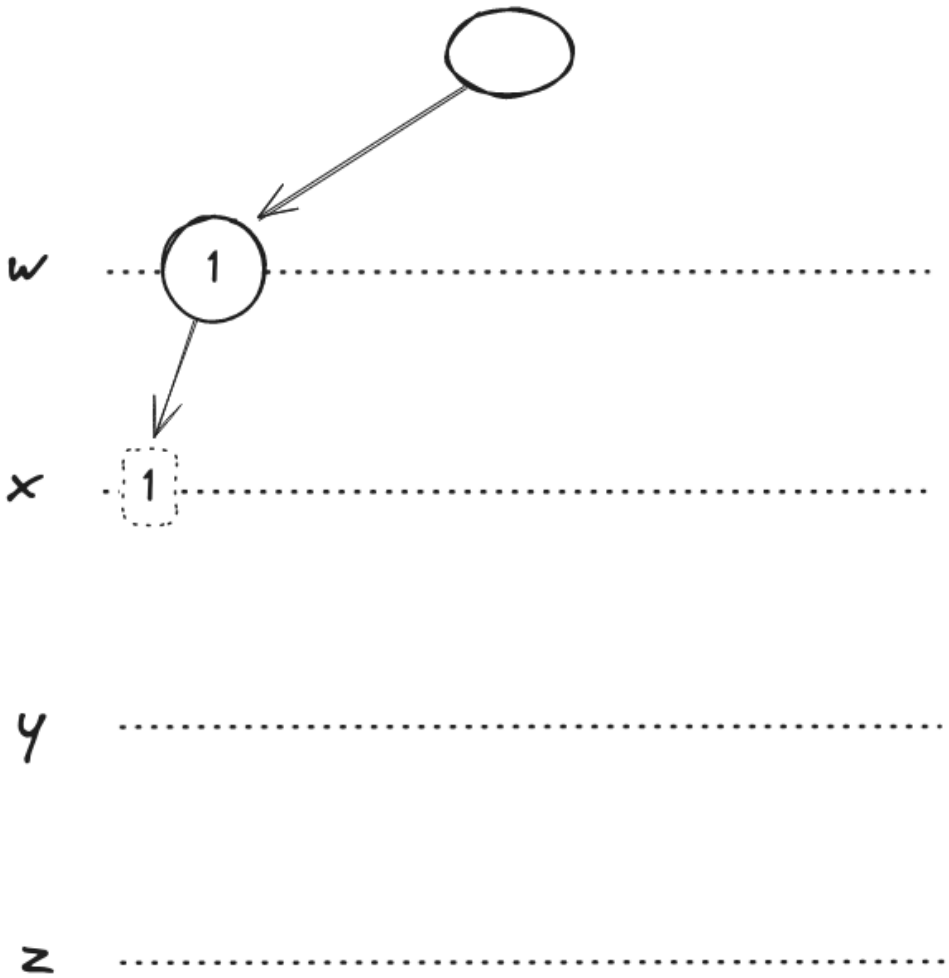
Constraints:

w = 2 * x

w < z

y > z

Backtracking Example



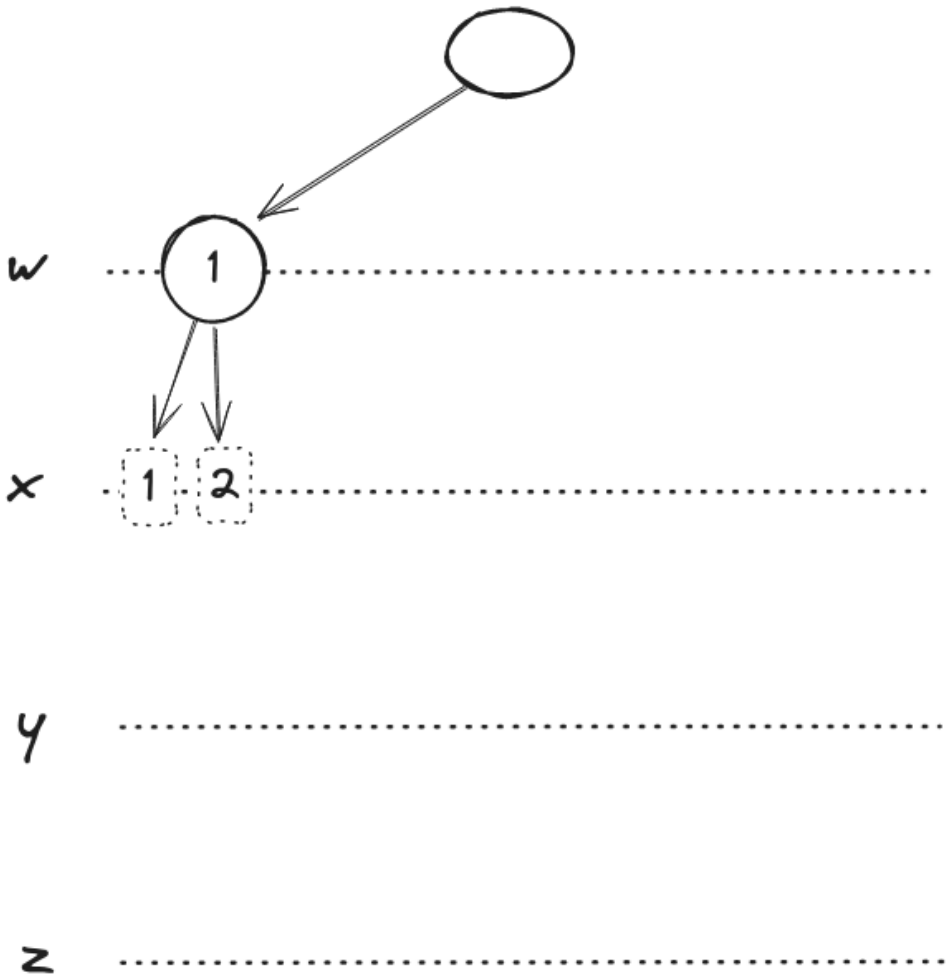
Constraints:

w = 2 * x

w < z

y > z

Backtracking Example



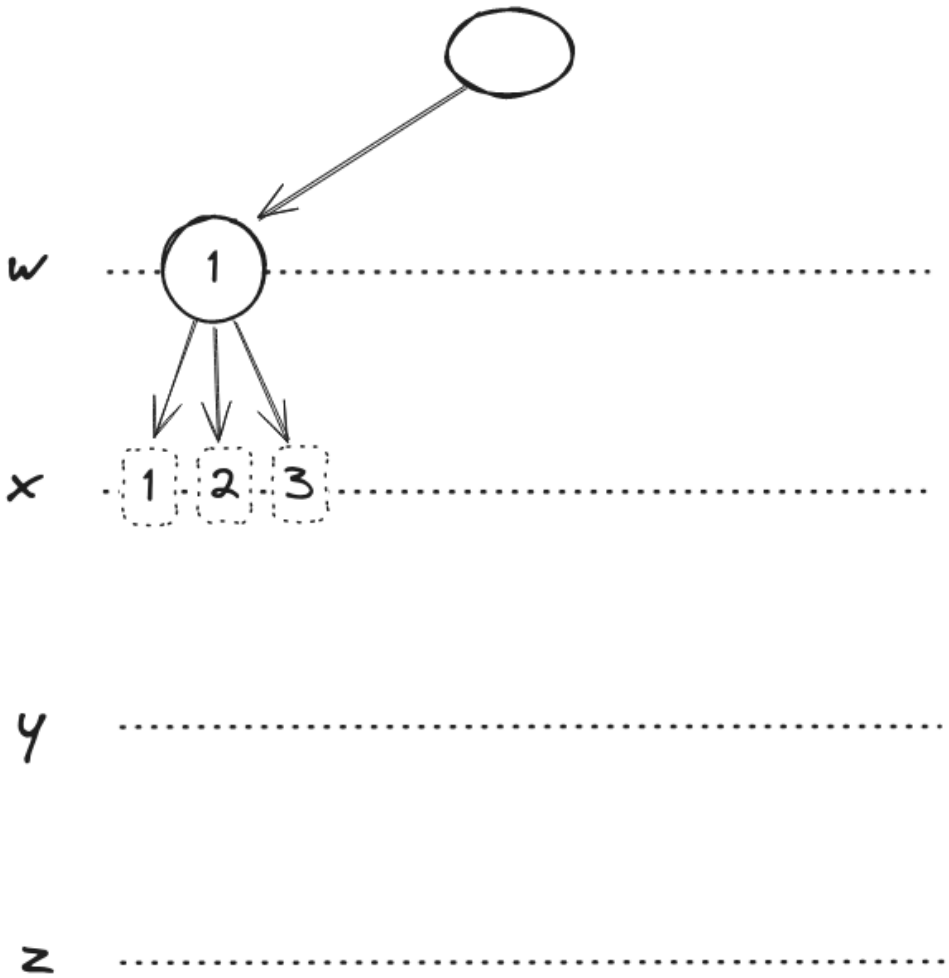
Constraints:

w = 2 * x

w < z

y > z

Backtracking Example



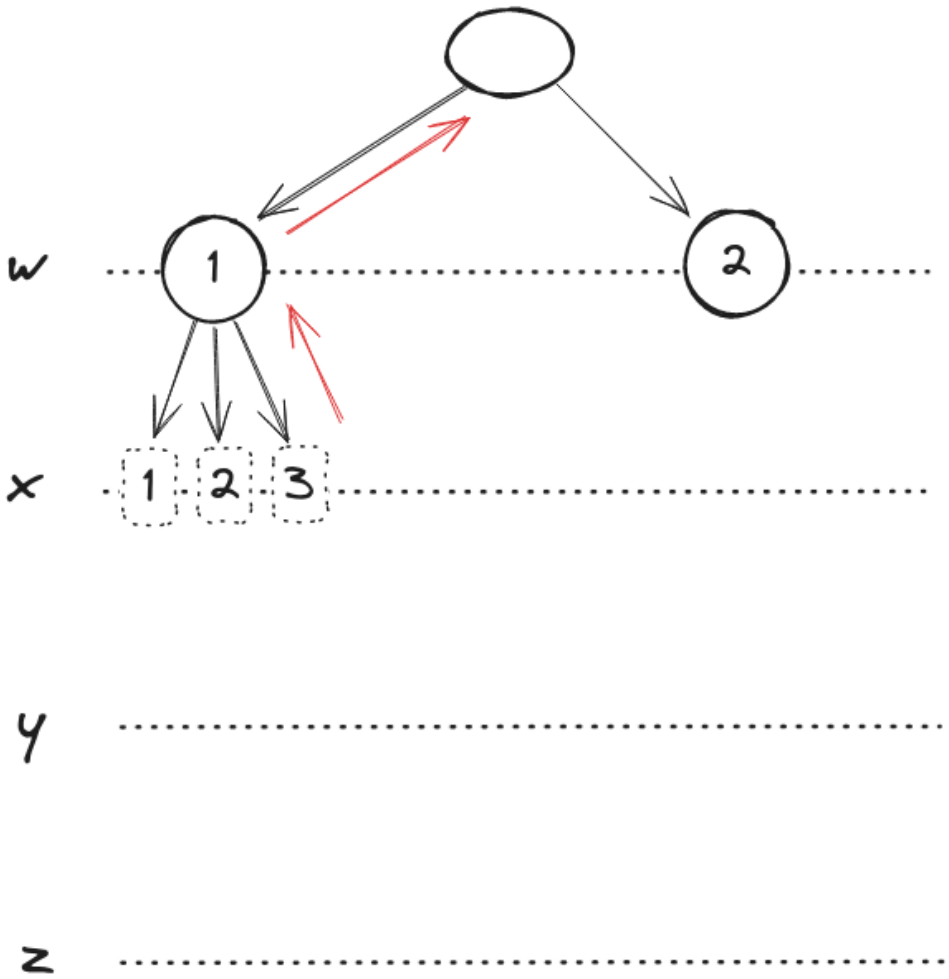
Constraints:

$w = 2 * x$

$w < z$

$y > z$

Backtracking Example



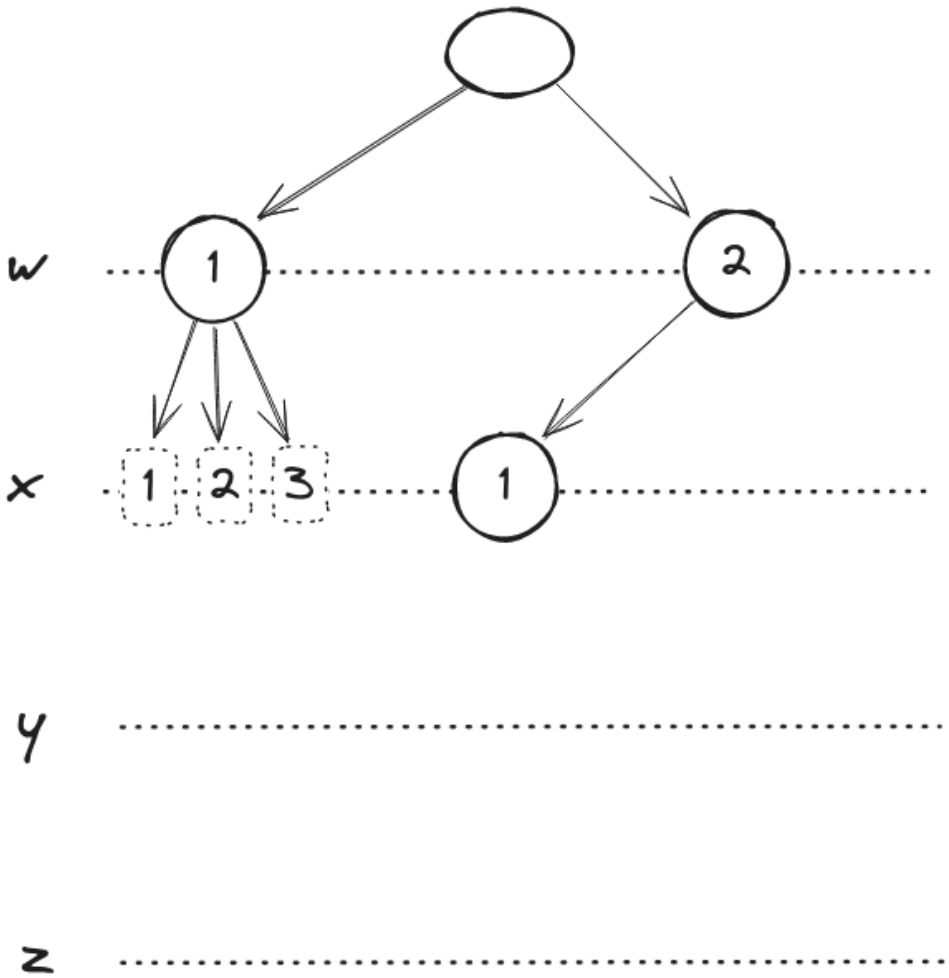
Constraints:

$w = 2 * x$

$w < z$

$y > z$

Backtracking Example



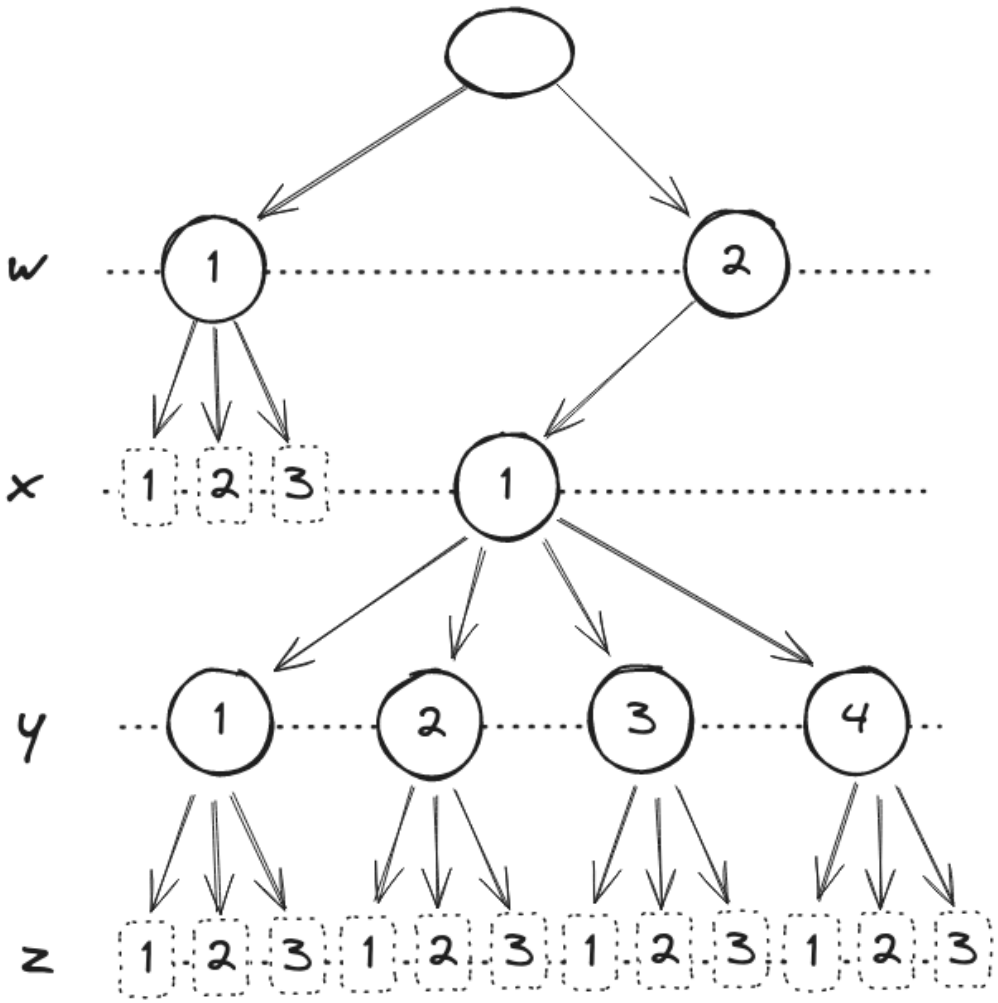
Constraints:

$w = 2 * x$

$w < z$

$y > z$

Backtracking Example



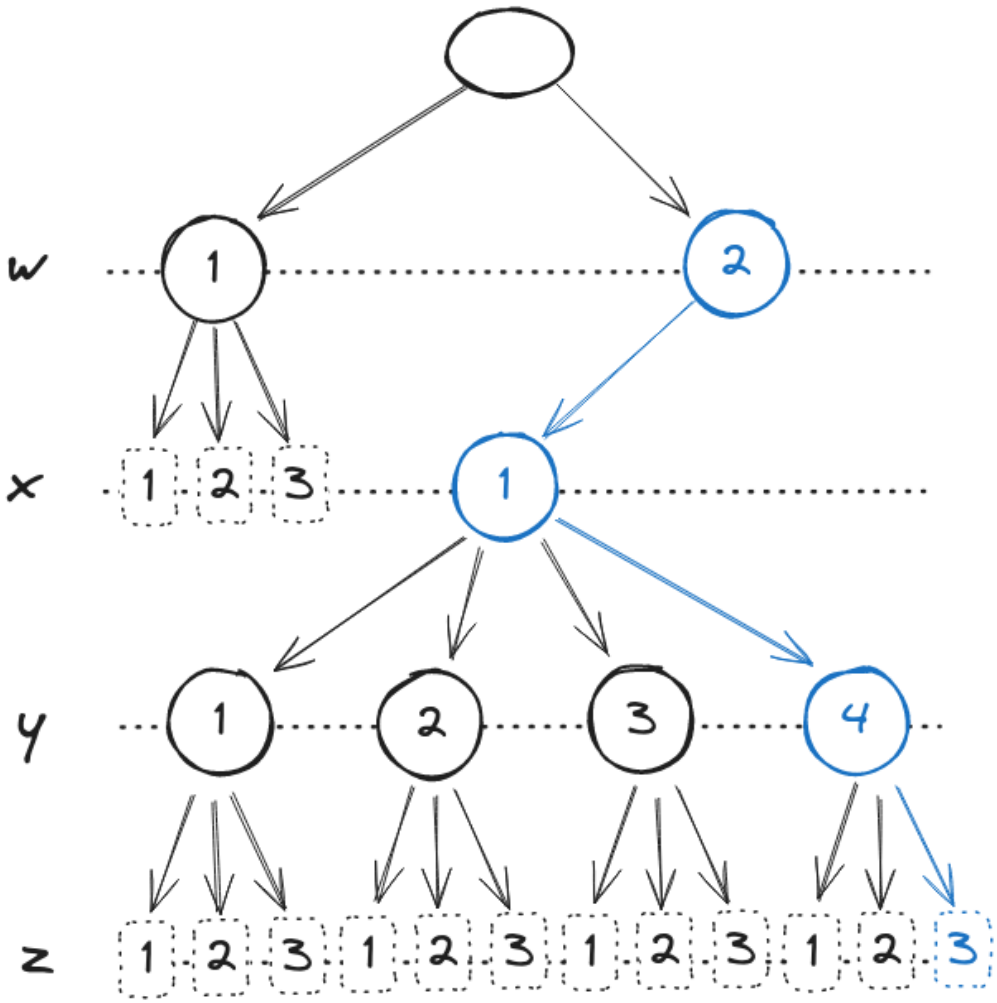
Constraints:

$w = 2 * x$

$w < z$

$y > z$

Backtracking Example



Constraints:

$w = 2 * x$

$w < z$

$y > z$

Solution:

$w = 2$

$x = 1$

$y = 4$

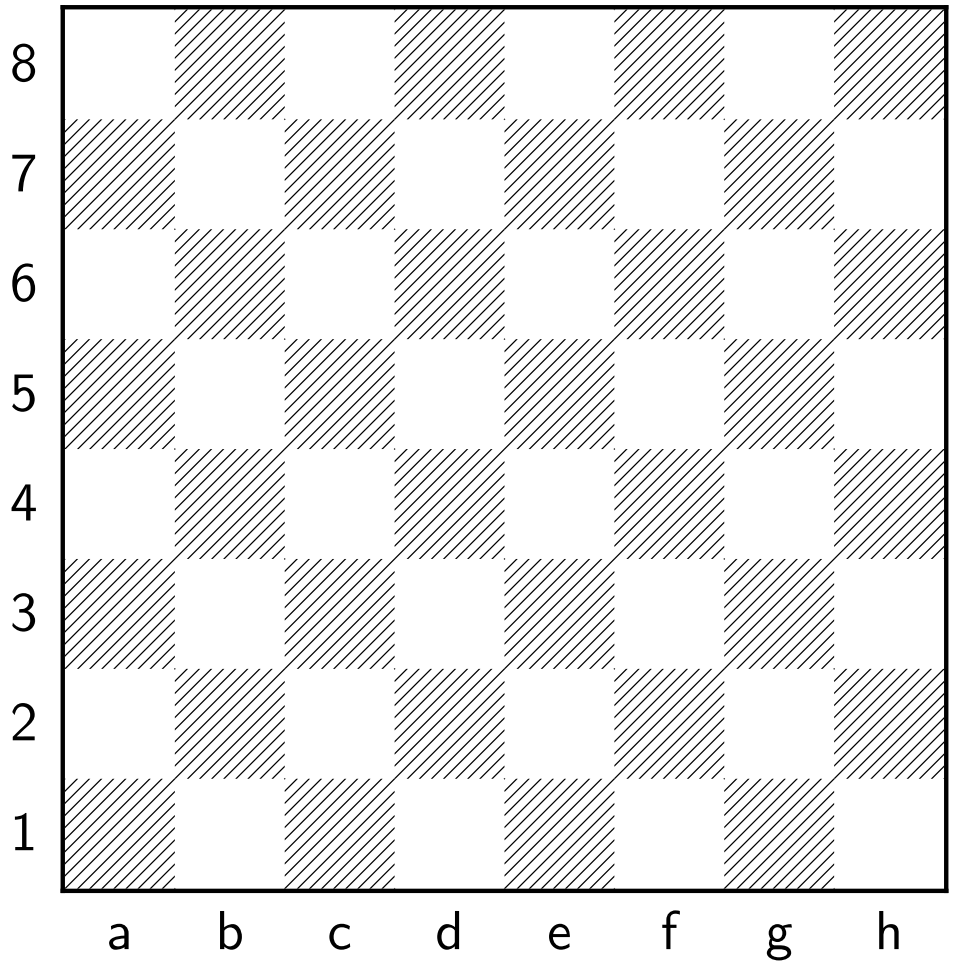
$z = 3$

Kinda like search...

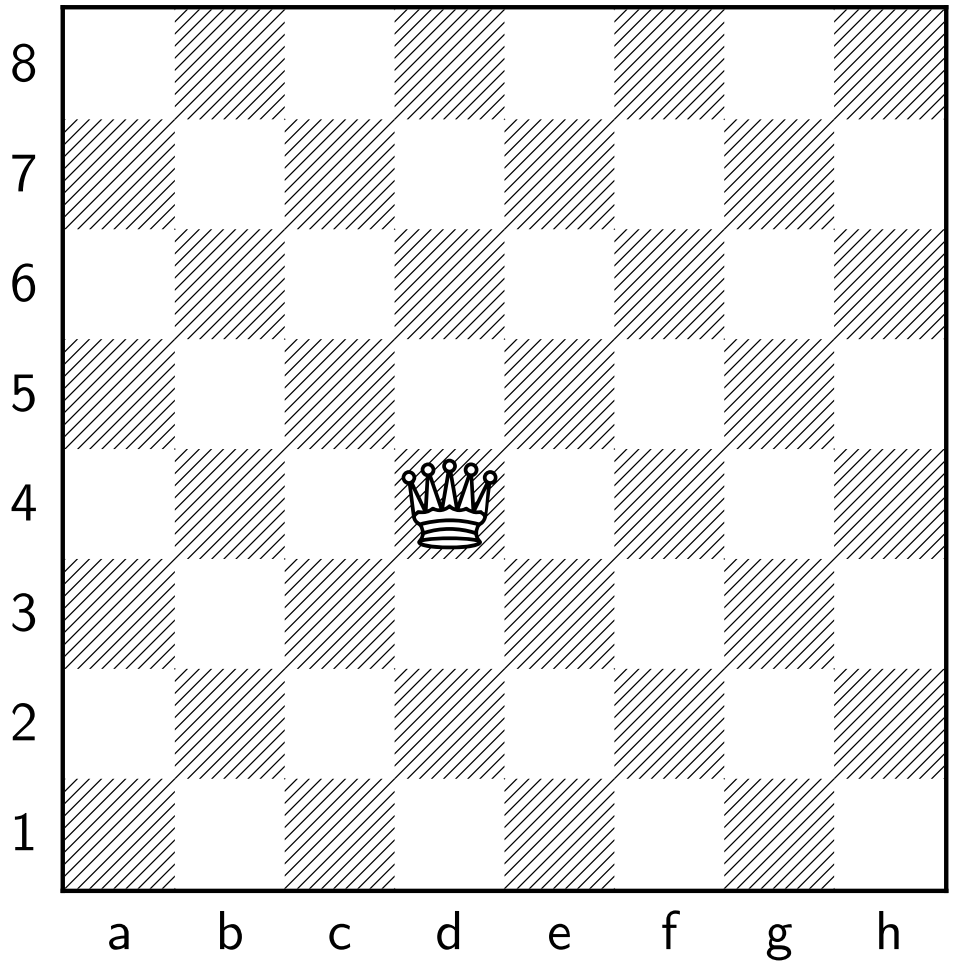
Can we do better?

8-Queens Problem

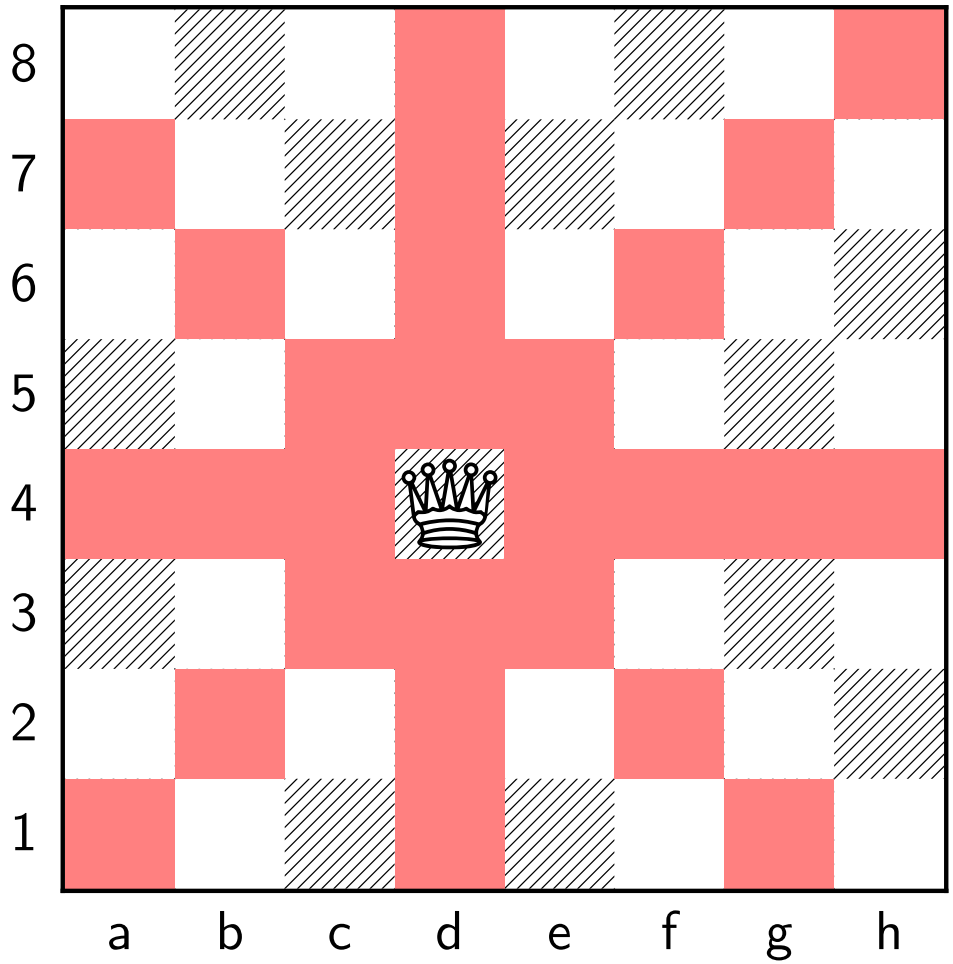
Chessboard



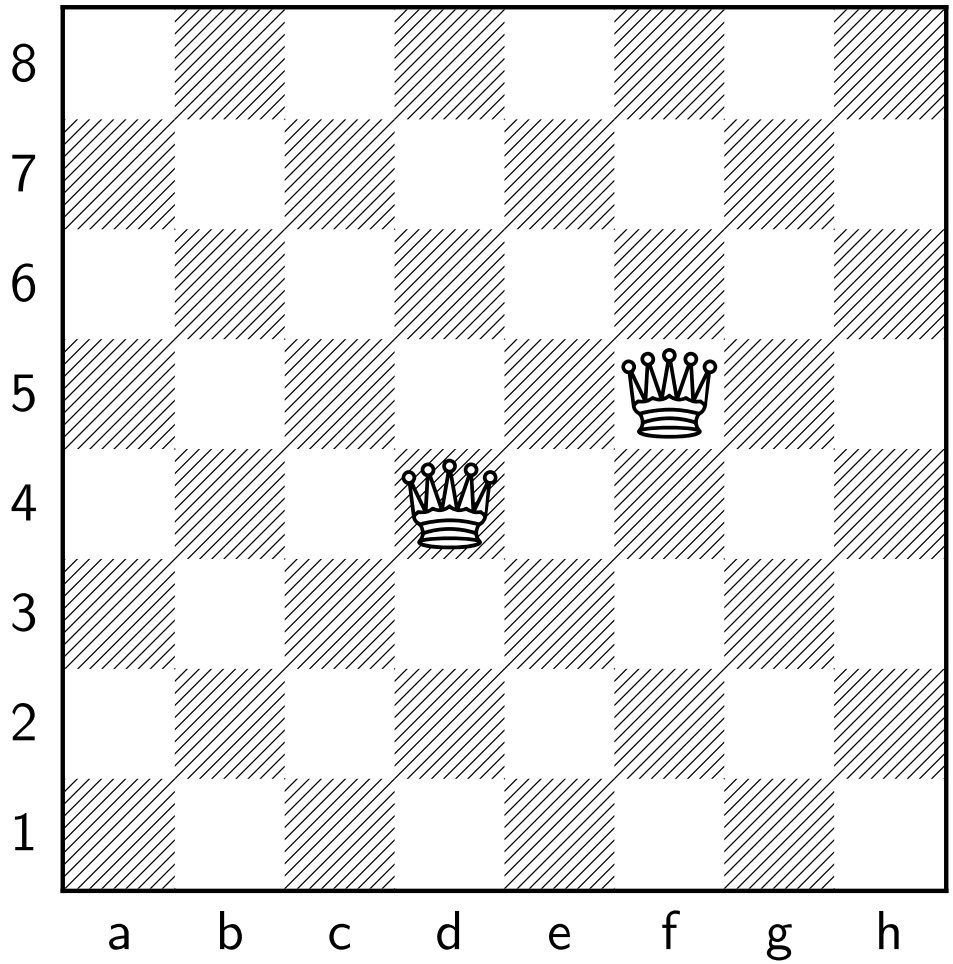
One Queen



One Queen



Two Queens



N-Queens Problem

- Scalable

N-Queens Problem

- Scalable
  - 8-Queens $\rightarrow$ N-Queens

N-Queens Problem

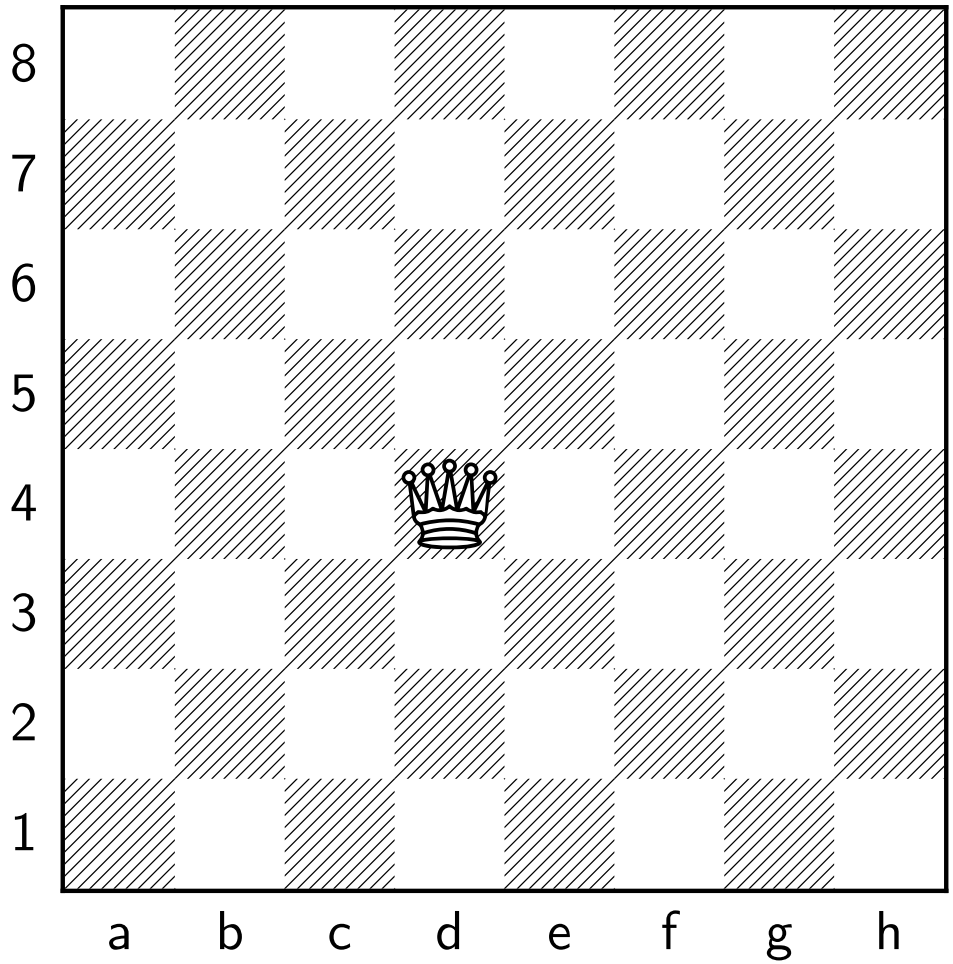
- Scalable
  - 8-Queens $\rightarrow$ N-Queens
  - $n \times n$ chessboard

N-Queens Problem

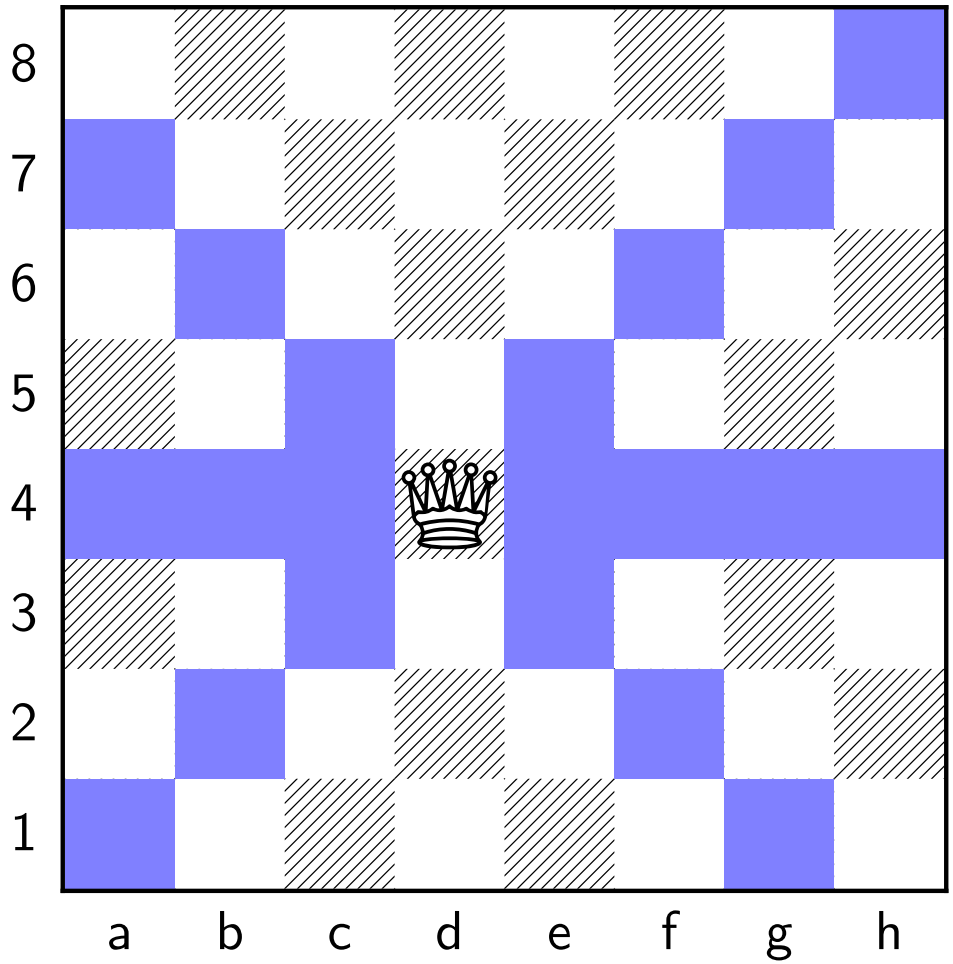
- Scalable
  - 8-Queens $\rightarrow$ N-Queens
  - $n \times n$ chessboard
  - n Queens

Inference

Inference Example



Inference Example



Improvements

- Inference

Improvements

- Inference
  - Forward Checking

Improvements

- Inference
  - Forward Checking
  - Arc consistency

Improvements

- Inference
  - Forward Checking
  - Arc consistency
    - AC-1

Improvements

- Inference
  - Forward Checking
  - Arc consistency
    - AC-1
    - AC-3

Improvements

- Inference
  - Forward Checking
  - Arc consistency
    - AC-1
    - AC-3
- Variable Ordering: fail early

Oxiflex

Demo

Constraint Programming Solver for MiniZinc written in Rust

Constraint Programming Solver for MiniZinc written in Rust

Solver

- Performance

Solver

- Performance
  - fast

Solver

- Performance
  - fast
  - like real fast

Solver

- Performance
  - fast
  - like real fast
- $\rightarrow$ no abstractions, no garbage collector, no JIT

Possible Language

Assembly

Possible Language

Assembly

or

Solver

- Performance
  - fast
  - like real fast

Solver

- Performance
  - fast
  - like real fast
- Correctness
  - Prevent bugs

Rust

Rust

- Performance
  - fast
  - like real fast
- Correctness
  - Prevent bugs

Rust

- Performance
  - fast
  - like real fast
- Correctness
  - Prevent bugs
- Ease of use

Rust

- Performance
  - fast
  - like real fast
- Correctness
  - Prevent bugs
- Ease of use
  - Library manager - Cargo

Rust

- Performance
  - fast
  - like real fast
- Correctness
  - Prevent bugs
- Ease of use
  - Library manager - Cargo
  - Functional

Constraint Programming Solver for MiniZinc written in Rust

Benchmarks

The end