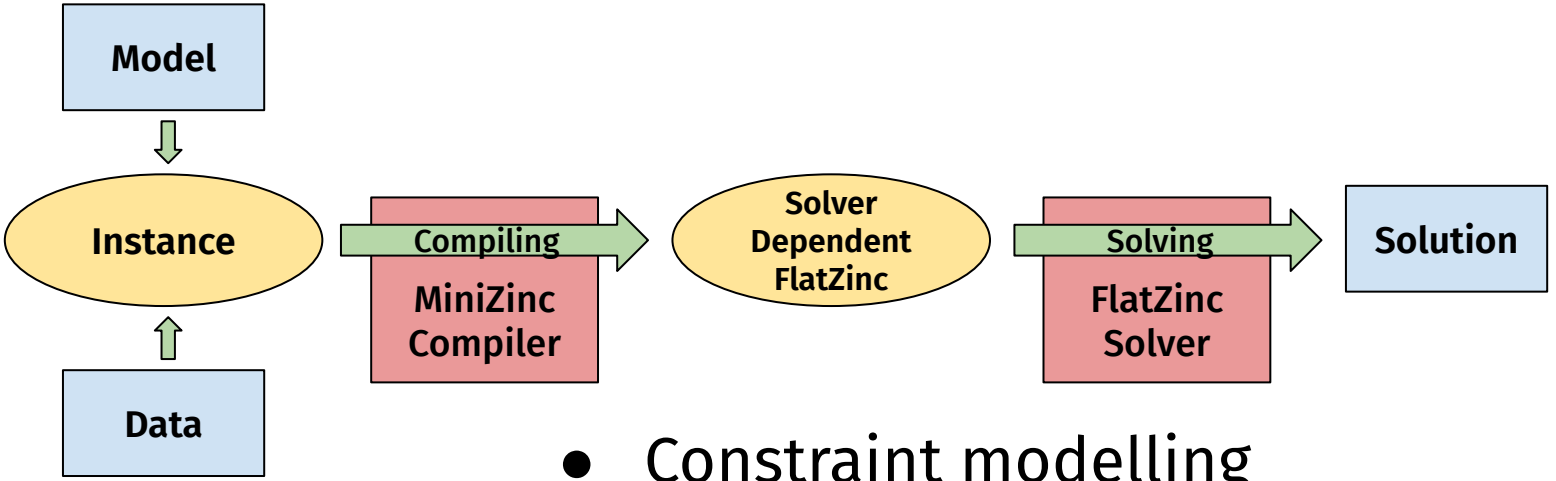
MONASH
University

An Abstract Machine Model for MiniZinc

Jip J. Dekker, Graeme Gange, Maria Garcia de la Banda, Andreas Schutt, Peter J. Stuckey, and Guido Tack

Introducing MiniZinc



- Constraint modelling
- Solver independent
- Solvers based on:
  CP, LCG, MIP, SAT, etc.

Term rewriting

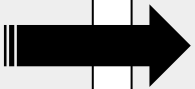
```
predicate two_apart(var int:i,
var int: j) =
  (i+2 = j) \/ (j+2 = i);
```

MiniZinc Library

```
var 1..5: x;
var 3..6: y;

constraint two_apart(x,y);
```

MiniZinc Model



```
var 1..5: x;
var 3..6: y;

constraint b1= (x+2 = y);
constraint b2= (y+2 = x);
constraint b1 \/ b2;
```

1 Step

Compiling MiniZinc

Compiling to **efficient FlatZinc** is complex:

- Compute all fixed expressions
- Avoid duplicate constraints
- Eliminate duplicate variables
- Remove unconnected constraints
- Provide tight variable domains

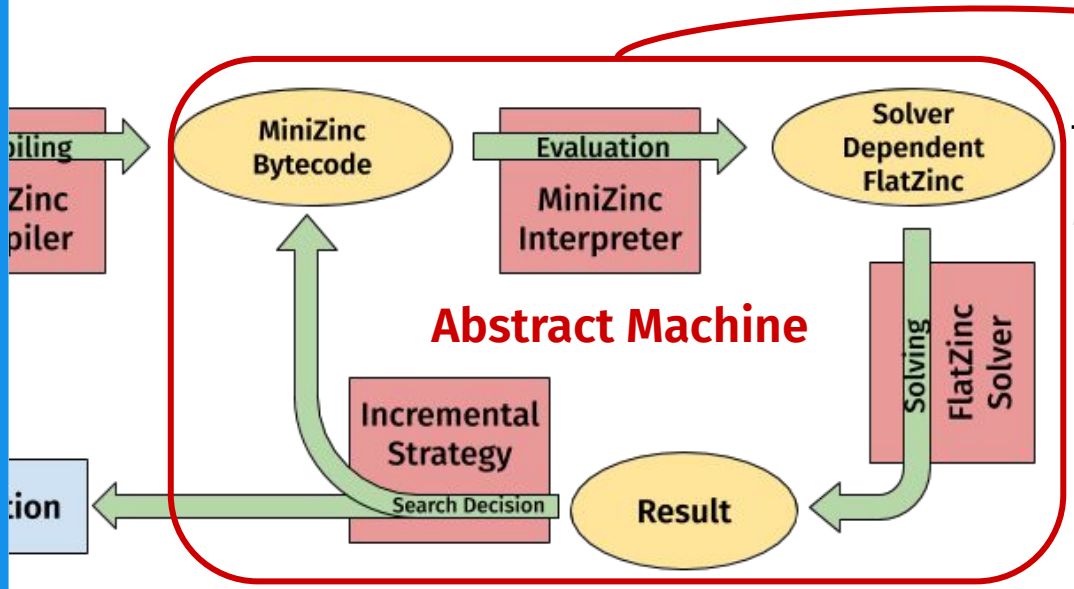
Incremental Search

Increased need for Incremental Search:

- Meta-Heuristics (LNS, Pareto, etc.)
- Interactive Optimisation
- Partial Evaluation

How can we introduce Incremental Search into MiniZinc?

In this paper



The incremental evaluation of MiniZinc without giving up:

- Tight variable domains
- Common Subexpression Elimination (CSE)
- Perfect Dependency Tracking

Thank you!

Jip J. Dekker

PhD Candidate @ Monash University
@ Data61, CSIRO

Incremental Search in MiniZinc

Contact me:

jip.dekker@monash.edu

@DekkerOne

@Dekker1