MiniZinc Basic Components

Jimmy Lee & Peter Stuckey

Overview

- Basic modeling features in MiniZinc
  - Types
  - Parameters
  - Decision Variables
  - Arithmetic Expressions
  - (Arithmetic) Constraints
  - Structure of a model

Types

Types available in MiniZinc are

- Integer `int` or range `1..n` or set of integers
  - `l..u` is integers {l, l+1, l+2, .., u}
- Floating point number `float` or range `1.0 .. f` or set of floats
- Boolean `bool`
- Strings `string` (only fixed strings)
- Arrays (see separate chapter)
- Sets (see separate chapter)

3

Variables

- Variables are critical for modeling a problem
- There are two types
  - Parameters
  - Decision variables
- Parameters reflect (fixed) information that is used to describe the problem (input)
- Decision variables are the decisions we wish to make to solve the problem (output)

4

Parameters

- Parameters are like variables in a standard programming language. They must be assigned a value (but only one)
- A parameter declaration has the form
  - [`par`] <type>: <varname> [= <exp>] `;`
  - where <type> is a type
  - <varname> is the name of the parameter
  - the `par` is optional and typically omitted
  - the optional <exp> gives a value for the parameter
    - it can instead be given a value later

5

Decision Variables

- Decision variables are like variables in mathematics. They are declared with a type and the var keyword. Their value is computed by a solver so that they satisfy the model
- A decision variable declaration takes the form
  - `var` <type>: <varname> [= <exp>] `;`
- Typically decision variables are not equated to an expression

6

Assignments

- An assignment is of the form
  - <varname> = <exp> ;
- It gives a value to the variable <varname>
- Assignments are used to
  - give a value to a parameter
  - define a decision variable name for an expression
- Examples
  - `int: m;  m = 3 * n;`
  - `var int: d; d = abs(x - y);`
- Assignments can be part of the declaration, e.g. `var int: d = abs(x - y);`

7

Instantiations

- Variables have an instantiation which specifies if they are parameters or decision variables
- The type + instantiation is called the type-inst
  - e.g. `var int, var 0..3, par float, par bool`
  - usually we omit par, e.g. `float` = `par float`
- MiniZinc errors are often couched in terms of mismatched type-insts

8

Comments

- Comments in MiniZinc files are
  - anything in a line after a `%`
  - anything between `/*` and `*/`
- (Just like in programming) It is valuable to
  - have a header comment describing the model at the top of the file
  - describe each parameter
  - describe each decision variable
  - and describe each constraint

9

Strings

Strings are provided for output

- An output item has the form

  `output` <list of strings>;
- String literals are like those in C:
  - enclosed in `"  "`
- They cant extend across more than one line
- Backslash for special characters \n \t etc
- Built in functions are
  - `show(v)`
  - `\(v)` show v inside a string literal
  - `"house"++"boat"` for string concatenation

10

Arithmetic Expressions

- MiniZinc provides the standard arithmetic operations
  - Floats: `*  /  +  -`
  - Integers: `*  div mod  +  -`
- Integer and float literals are like those in C
- There is automatic coercion from integers to floats. The builtin `int2float`(intexp) can be used to explicitly coerce them
- Builtin arithmetic functions:
  - `abs, sin, cos, atan, ...`

11

Constraints

- Basic arithmetic constraints are built using the arithmetic relational operators are
  - `=  !=  >  <  >=  <=`
- Constraints in MiniZinc are written in the form
  - `constraint` <constraint-expression>
- Examples
  - `constraint x <= y;`
  - `constraint x + 2*y - abs(z) != 0;`

12

Basic Structure of a Model

- A MiniZinc model is a sequence of items
- The order of items does not matter
- The kinds of items are
  - An inclusion item
    - `include` <filename (which is a string literal)>;
  - An output item
    - `output` <list of string expressions>;
  - A variable declaration
  - A variable assignment
  - A constraint
    - `constraint` <Boolean expression>;

13

Basic Structure of a Model

- The kinds of items (cont.)
  - A solve item (a model must have exactly one of these)

```
solve satisfy;
solve maximize <arith. expression>;
solve minimize <arith. expression>;
```

  - Predicate, function and test items
  - Annotation items
- Identifiers in MiniZinc start with a letter followed by other letters, underscores or digits
- In addition, the underscore `_' is the name for an anonymous decision variable

14