MiniZinc Basic Components

Peter Stuckey

Overview

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

Parameters

In MiniZinc there are two kinds of variables:

Parameters-These are like variables in a standard programming language. They must be assigned a value (but only one).

They are declared with a type (or a range/set).

You can use par but this is optional

The following are logically equivalent

```
 int: i=3;
par int: i=3;
 int: i;  i=3;
```

Decision Variables

Decision variables-These 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.

Typically they are declared using a range or a set rather than a type name

The following are logically equivalent

```
var int: i; constraint i >= 0; constraint i <= 4;
var 0..4: i;
var {0,1,2,3,4}: i;
```

Types

Allowed types for variables 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` (but these cannot be decision variables)
- Arrays
- Sets

Instantiations

Variables have an instantiation which specifies if they are parameters or decision variables.

The type + instantiation is called the type-inst.

MiniZinc errors are often couched in terms of mismatched type-insts…

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

Strings

Strings are provided for output

- An output item has form

  `output` <list of strings>;

- String literals are like those in C:
  - enclosed in ″ ″
- They cannot 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

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,...`

Constraints

- Basic arithmetic constraints are built using the arithmetic relational operators are

  `==    !=     >    <    >=    <=`

- Constraints in MiniZinc are written in the form

  `constraint` <constraint-expression>

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>;

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

EOF