Introduction to the MiniZinc Toolchain

Frej Knutar Lewander and Pierre Flener

Modelling for Combinatorial Optimisation (course 1DL451)
&
Combinatorial Optimisation and Constraint Programming (course 1DL442)
at
Uppsala University, Sweden

24th August 2023

Help Sessions

- Scheduled time for working on an assignment or your project and asking questions.
- Help sessions are *not* labs: *you have to work on the current assignment or your project also before and after each help session!*
- The assignments and project are challenging (no exam!) and will require much time.

Assignments and Project

- There are three assignments and one project.
- Read their *Submission Instructions* and *Grading Rules* very carefully, else you may waste a lot of time, if not lose points needlessly.

  In particular, the grading rule

  If ..., **and** ..., **and** ..., **then** you score ... points, **else** ...

  obviously means that you must fulfil *every* condition to qualify for the **then** branch.
- You upload your PDF report (singular) and MiniZinc models via Studium.

MiniZinc

The important parts:

- It is a declarative language!
- Do not try and design algorithms in order to solve the problems!
- It takes some time to get used to declarative modelling.
- Read the course slides and the MiniZinc Handbook (https://www.minizinc.org/doc-latest/index-en.html).

MiniZinc

Designing MiniZinc models can feel strange, confusing, and even frustrating during the first few weeks, but only until you get into the declarative mindset, so do not panic!

For example, a common cause for confusion is that there are no loops in MiniZinc!

The `forall` keyword is the logical universal quantifier ($\forall$).
For example, the constraint

```
constraint forall(i in 1..10)(X[i] < y)
```

means

MiniZinc

Designing MiniZinc models can feel strange, confusing, and even frustrating during the first few weeks, but only until you get into the declarative mindset, so do not panic!

For example, a common cause for confusion is that there are no loops in MiniZinc!

The `forall` keyword is the logical universal quantifier ($\forall$).
For example, the constraint

```
constraint forall(i in 1..10)(X[i] < y)
```

means $\forall i \in \{1, 2, \ldots, 10\} : X_i < y$

MiniZinc

Designing MiniZinc models can feel strange, confusing, and even frustrating during the first few weeks, but only until you get into the declarative mindset, so do not panic!

For example, a common cause for confusion is that there are no loops in MiniZinc!

The `forall` keyword is the logical universal quantifier ($\forall$).
For example, the constraint

```
constraint forall(i in 1..10)(X[i] < y)
```

means $\forall i \in \{1, 2, \ldots, 10\} : X_i < y$,

which is the same as $\bigwedge_{i=1}^{10} X_i < y$

MiniZinc

Designing MiniZinc models can feel strange, confusing, and even frustrating during the first few weeks, but only until you get into the declarative mindset, so do not panic!

For example, a common cause for confusion is that there are no loops in MiniZinc!

The `forall` keyword is the logical universal quantifier ($\forall$).
For example, the constraint

```
constraint forall(i in 1..10)(X[i] < y)
```

means $\forall i \in \{1, 2, \ldots, 10\} : X_i < y$,
which is the same as $\bigwedge_{i=1}^{10} X_i < y$, that is $(X_1 < y) \wedge (X_2 < y) \wedge \cdots \wedge (X_{10} < y)$.

MiniZinc

Designing MiniZinc models can feel strange, confusing, and even frustrating during the first few weeks, but only until you get into the declarative mindset, so do not panic!

For example, a common cause for confusion is that there are no loops in MiniZinc!

The `forall` keyword is the logical universal quantifier ($\forall$).
For example, the constraint

```
constraint forall(i in 1..10)(X[i] < y)
```

means $\forall i \in \{1, 2, \ldots, 10\} : X_i < y$,
which is the same as $\bigwedge_{i=1}^{10} X_i < y$, that is $(X_1 < y) \wedge (X_2 < y) \wedge \cdots \wedge (X_{10} < y)$.

What does `constraint (x < y) <-> (forall(i in N)(Z[i]>x))` mean?

MiniZinc – Comprehensions

Because there are no loops, *array comprehensions* and *set comprehensions* are the bread and butter of MiniZinc.

MiniZinc – Comprehensions

Because there are no loops, *array comprehensions* and *set comprehensions* are the bread and butter of MiniZinc.

They are similar to set comprehensions in mathematics, such as $\{i^2 \mid i \in \{1, 2, \ldots, 10\},\ i \equiv 0 \mod 2\}$.

MiniZinc – Comprehensions

Because there are no loops, *array comprehensions* and *set comprehensions* are the bread and butter of MiniZinc.

They are similar to set comprehensions in mathematics, such as $\{i^2 \mid i \in \{1, 2, \ldots, 10\},\ i \equiv 0 \mod 2\}$.

In MiniZinc: `{i*i | i in 1..10 where i mod 2 = 0}`

MiniZinc – Comprehensions

Because there are no loops, *array comprehensions* and *set comprehensions* are the bread and butter of MiniZinc.

They are similar to set comprehensions in mathematics, such as $\{i^2 \mid i \in \{1, 2, \ldots, 10\},\ i \equiv 0 \mod 2\}$.

In MiniZinc: `{i*i | i in 1..10 where i mod 2 = 0}`

Array (or: list) comprehensions also exist in Python:

```
>>> [i*i for i in range(1,11) if i%2 == 0]
[4, 16, 36, 64, 100]
```

MiniZinc – Comprehensions

Note that

```
constraint forall(i in 1..10)(X[i] < y);
```

is actually syntactic sugar for:

```
constraint forall([X[i] < y | i in 1..10]);
```

So everything really is just array comprehensions!

MiniZinc – Comprehensions

To get you well started with MiniZinc, its syntax, and comprehensions, there are some warm-up exercises (no hand-in!) at https://user.it.uu.se/~pierref/courses/COCP/assignments/exercises/comprehensions.pdf
(there are links to them from the *Assignments* and *Resources* subpages of the course page https://user.it.uu.se/~pierref/courses/COCP/course.html).

MiniZinc

In the beginning, you will have a lot of questions and may get stuck due to syntax: make sure to ask during help sessions!

MiniZinc Toolchain

- A key feature of the course is that you *must* use and evaluate backends — namely `Gecode`, Google's `CP-SAT` (or `Chuffed`), `Gurobi` (or `COIN-BC` or `HiGHS`), `Yuck`, and `PicatSAT` — of five solving technologies.

MiniZinc Toolchain

- A key feature of the course is that you *must* use and evaluate backends — namely `Gecode`, Google's `CP-SAT` (or `Chuffed`), `Gurobi` (or `COIN-BC` or `HiGHS`), `Yuck`, and `PicatSAT` — of five solving technologies.
- But the MiniZinc toolchain only bundles `Gecode`, `Chuffed`, `COIN-BC`, and `HiGHS`, which are of only three of those five solving technologies.

MiniZinc Toolchain

- A key feature of the course is that you *must* use and evaluate backends — namely `Gecode`, Google's `CP-SAT` (or `Chuffed`), `Gurobi` (or `COIN-BC` or `HiGHS`), `Yuck`, and `PicatSAT` — of five solving technologies.
- But the MiniZinc toolchain only bundles `Gecode`, `Chuffed`, `COIN-BC`, and `HiGHS`, which are of only three of those five solving technologies.
- Installing the other backends of this course is a bit tricky, different for each operating system, and time-consuming. We do not expect you to do this.

MiniZinc Toolchain

- A key feature of the course is that you *must* use and evaluate backends — namely `Gecode`, Google's `CP-SAT` (or `Chuffed`), `Gurobi` (or `COIN-BC` or `HiGHS`), `Yuck`, and `PicatSAT` — of five solving technologies.
- But the MiniZinc toolchain only bundles `Gecode`, `Chuffed`, `COIN-BC`, and `HiGHS`, which are of only three of those five solving technologies.
- Installing the other backends of this course is a bit tricky, different for each operating system, and time-consuming. We do not expect you to do this.
- Each run of the experiments of each assignment can take up to 1 CPU day!

MiniZinc Toolchain

- A key feature of the course is that you *must* use and evaluate backends — namely `Gecode`, Google's `CP-SAT` (or `Chuffed`), `Gurobi` (or `COIN-BC` or `HiGHS`), `Yuck`, and `PicatSAT` — of five solving technologies.
- But the MiniZinc toolchain only bundles `Gecode`, `Chuffed`, `COIN-BC`, and `HiGHS`, which are of only three of those five solving technologies.
- Installing the other backends of this course is a bit tricky, different for each operating system, and time-consuming. We do not expect you to do this.
- Each run of the experiments of each assignment can take up to 1 CPU day!
- You can connect to one of our Linux computers via `ssh` and run everything remotely with all five solving technologies (while you sleep) using our experiment script, which even generates and integrates result tables into the LaTeX demo report!

Your Own Computer vs Our Linux Computers

Designing models on your own computer (without two of the considered technologies) and running (only the final) experiments remotely on all five technologies is not good:

Your Own Computer vs Our Linux Computers

Designing models on your own computer (without two of the considered technologies) and running (only the final) experiments remotely on all five technologies is not good: you ideally want to test *all* five technologies *while* you are designing your model.

Your Own Computer vs Our Linux Computers

Designing models on your own computer (without two of the considered technologies) and running (only the final) experiments remotely on all five technologies is not good: you ideally want to test *all* five technologies *while* you are designing your model.

Indeed, different models for the *same* problem can give *very* different performance:

Model	`Gecode`	...	`Gurobi`	...	`PicatSAT`
A	`timeout`	...	`timeout`	...	40*s*
B	5*s*	...	`timeout`	...	350*s*
C	`timeout`	...	`timeout`	...	`timeout`

Your Own Computer vs Our Linux Computers

Designing models on your own computer (without two of the considered technologies) and running (only the final) experiments remotely on all five technologies is not good: you ideally want to test *all* five technologies *while* you are designing your model.

Indeed, different models for the *same* problem can give *very* different performance:

Model	Gecode	...	Gurobi	...	PicatSAT
A	timeout	...	timeout	...	40*s*
B	5*s*	...	timeout	...	350*s*
C	timeout	...	timeout	...	timeout

If you designed model A, then how would you know if it is good or not?
If you designed model C, then how would you know that you need to rethink?

Your Own Computer vs Our Linux Computers

That was an extreme example, but not unreasonable.

In general, you should be careful.

For this course, as long as you *initially* experiment on at least the bundled `Gecode`, `Chuffed`, and `HiGHS` (or `COIN-BC`), you should be fine, but the *final* experiments must be on backends of *all* the five solving technologies of the course.

If it is fast enough, then consider an `ssh -X` session on one of our Linux computers!

MiniZinc Toolchain

In short:

- Install the MiniZinc toolchain on your own computer (if you have one) from https://www.minizinc.org/software.html.

MiniZinc Toolchain

In short:

- Install the MiniZinc toolchain on your own computer (if you have one) from https://www.minizinc.org/software.html.
- Run your experiments (remotely via `ssh`) on one of our Linux computers.

MiniZinc Toolchain

In short:

- Install the MiniZinc toolchain on your own computer (if you have one) from `https://www.minizinc.org/software.html`.
- Run your experiments (remotely via `ssh`) on one of our Linux computers.
- Optionally run our MiniZinc integrated development environment (IDE) remotely via `ssh -X` on one of our Linux computers.

Running MiniZinc on Our Linux Computers

You can connect to one of our Linux computers by using `ssh` in a terminal on your own macOS or Linux computer, say:

```
> ssh -X username@siegbahn.it.uu.se
```

where `username` is your UU account name and `>` is the prompt (and hence not part of the actual command).

You are asked to enter a password: use your password A.

Using the optional `-X` option requires an X server, such as Xming or XQuartz.

Running MiniZinc on Our Linux Computers

You can connect to one of our Linux computers by using `ssh` in a terminal on your own macOS or Linux computer, say:

```
> ssh -X username@siegbahn.it.uu.se
```

where `username` is your UU account name and `>` is the prompt (and hence not part of the actual command).

You are asked to enter a password: use your password A.

Using the optional `-X` option requires an X server, such as Xming or XQuartz.

If you are under Windows, then have a look at "Windows Subsystem for Linux" (**wsl2**), in order to install and run Linux as a subsystem of Windows (not as a virtual machine).

Running MiniZinc on Our Linux Computers

If you are connected to one of our Linux computers through `ssh`,
then you must (once!) perform an installation for your account as follows:

```
> cd /it/kurs/consprog/minizinc/
> ./first_install.sh
```

You now have three new aliases in your `~\.bashrc` file, but before you can use any of them, you must for each already opened terminal reload the `~\.bashrc` file:

```
> source ~\.bashrc
```

Running MiniZinc on Our Linux Computers

If you are connected to one of our Linux computers through `ssh`,
then you must (once!) perform an installation for your account as follows:

```
> cd /it/kurs/consprog/minizinc/
> ./first_install.sh
```

You now have three new aliases in your `~\.bashrc` file, but before you can use any of them, you must for each already opened terminal reload the `~\.bashrc` file:

```
> source ~\.bashrc
```

To launch our MiniZinc command-line interface (CLI), use the following new alias:

```
> minizinc --help
```

If you are connected to one of our Linux computers through `ssh -X`,
then you can launch our MiniZinc IDE using the following new alias:

```
> minizinc-ide
```

Running MiniZinc on Our Linux Computers

Everything you need to know for the third new alias, namely `run_backends`, which automates all your experiments and generates and integrates a result table into the LaTeX demo report, can be found in https://user.it.uu.se/~pierref/courses/COCP/assignments/cheatsheet.pdf.

Read this document as soon as possible!

MiniZinc – Python Interface

There is a Python interface for MiniZinc
(see https://minizinc-python.readthedocs.io/en/latest) that may come in handy for your upcoming project, but it is *not* needed for the 3 assignments.

Bugs

- Some backends may turn out to have bugs:
  please tell us if you suspect having found one!

- Our cheatsheet will be updated, if need be, during the course:
  see https://user.it.uu.se/~pierref/courses/COCP/assignments/cheatsheet.pdf.

Locate Everything

Take a few minutes to locate (and bookmark) everything:

- The course homepage is at
  https://user.it.uu.se/~pierref/courses/COCP/course.html.
- The MiniZinc Handbook is at
  https://www.minizinc.org/doc-latest/index-en.html.
- Our cheatsheet is at
  https://user.it.uu.se/~pierref/courses/COCP/assignments/cheatsheet.pdf.
- The exercises on comprehensions are at
  https://user.it.uu.se/~pierref/courses/COCP/assignments/exercises.
- The assignments and the project guidelines are at
  https://user.it.uu.se/~pierref/courses/COCP/assignments and
  https://user.it.uu.se/~pierref/courses/COCP/project.html.

First Steps

What to do now:

- Look at the MiniZinc Handbook.
- Look at our exercises on comprehensions.
- Look at Assignment 1.
- Look at our cheatsheet.
- Install the MiniZinc toolchain on your own computer (if you have one).
- Connect remotely to one of our Linux computers.

Questions?