Instructions for using MiniZinc reformulation of time-series constraints with different solvers, e.g., SICStus Prolog, Choco, etc.

Installing MiniZinc 2.0:

1. http://www.minizinc.org/2.0/index.html

2. For future work with solvers a user needs to set up the PATH variable for the mzn2fzn file, which is normally distributed together with MiniZinc 2.0

For all solvers:

A MiniZinc model is a file with the 'mzn' extension, in which a user has to consult the 'time_series_constraints.mzn' file if they are going to use time-series constraints.

- A model may call time-series constraints, e.g., functional constraints, predicate constraints, footprint constraints.
- The format of the predicates for time-series constraints in MiniZinc is described in ‘time_series_constraints.mzn’.
- All the parameters, namely the feature, the pattern, the aggregator must be declared in the model. They can be assigned in file with the model, or later when the model is uploaded to the solver.
- An array of time-series decision variables must be declared.
- A model must have a 'solve' item.
- More information on MiniZinc models
http://www.minizinc.org/downloads/doc-latest/minizinc-tute.pdf
- Example of a model is in the attached file ‘tester_function.mzn’

SICStus Prolog 4.3.3

1. Preparing for work with MiniZinc
https://sicstus.sics.se/sicstus/docs/4.3.2/html/sicstus/MiniZinc.html

2. In SICStus to load a MiniZinc model:
mzn_run_file(‘MiniZincModelName.mzn’, [parameters([Parameters]), variables([Variables])], FZState).

where

- 'Parameters' is a list of pairs of the form "param = value", the parameter with the name "param" must appear in the MiniZinc model and it is value must be unassigned, otherwise it may lead to an infeasible problem.

- 'Variables' is a list of pairs of the form "name = variable", where 'name' is the name of variable that appears in the MiniZinc model and 'variable' is a SICStus variable.

- FZState is an unassigned at this stage SICStus variable, used for MiniZinc output if any.

Example:

mzn_run_file('tester_function.mzn', [parameters([aggregator = max, feature = max, pattern = peak, domain_lb = 1, domain_ub = 2, time_series_length = 10]), variables([x=X, result=Result])], FZState).

'X' is a list of time-series variables, a list of SICStus variables, after a call of this predicate, max_max_peak constraint is imposed on a time series of length 10 with every variable ranging over [1,2].

'Result' is the resulting variable, a SICStus variable.

We can also upload the values of parameters from a file with the '.dzn' extension:

mzn_run_file('tester_function.mzn', [data_file('max_max_peak.dzn'), variables([x=X, result=Result])]).

A user can add any additional constraints on X variables and then find solutions or maximise/ minimise some.

Example:

findall(solution(X,Result), (labeling([], X)), Solutions).

Solutions is a list of terms of the form solution(X,Result), where X is a time series and Result is the value of the maximum of the maxima of the peaks in this time series.

Choco 3.3.2:

1. Preparing for work with MiniZinc
Download 'choco-parsers-3.3.2-with-dependencies.jar'
from https://github.com/chocoteam/choco-parsers/releases

2. To call in the terminal

mzn2fzn PathToMiniZincModel -d ParameterFile

% Choco works with flatzinc code, thus we first need to convert a minizinc model to
% the flatzinc one.
% The PATH variable should be set up properly for the 'mzn2fzn' converter.

% 'PathToMiniZincModel' is the path to a MiniZinc model, i.e., to a file with the 'mzn'
% extensions

% 'ParameterFile' is the path to a file with the 'dzn' extension which contains parameters % of the model, if there are some not defined in the model.
% Example of a file with parameters 'max_max_peak' is attached.

% This call will produce a file with the same name as the file with the MiniZinc model but % with the ‘fzn’ extension.

java -server -cp .:PathToChoco chocosolver.parser.flatzinc.ChocoFZN PathToFZNModel -a

% ‘PathToChoco’ is the path to choco-parsers-3.3.2-with-dependencies.jar

% ‘PathToFZNModel’ is the path to the FlatZinc model that is the output file of the
% MiniZinc ->FlatZinc converter

% Option "-a" is for searching for all solutions