Step 2: What about Globals?

Institute for Software & Systems Engineering

What if want to use global constraints as soft constraints? Consider (`reified.mzn`):

```
include "globals.mzn";
include "soft_constraints/soft_constraints.mzn";
array[1..5] of var 0..5: x;
constraint alldifferent(x); % hard constraint

% but it would be nice to have x increase ...
nScs = 1; penalties = [1];
constraint increasing(x) <-> satisfied[1];

solve maximize bool2int(satisfied[1]);
output["satisfied[1] = \(satisfied[1]), x = \(x)"];
```

Solved with Gecode:

```
MiniZinc: flattening error: 'increasing_int' is used in a reified
context but no reified version is available
```

Step 2: What about Globals?

Institute for Software & Systems Engineering

a) Use a solver that has the reified global, e.g., here:
   - G12-FD
   - JaCoP

b) Use a decomposition from the MiniZinc standard library (`reified_fix.mzn`)

```
include "globals.mzn";
include "soft_constraints/soft_constraints.mzn";
array[1..5] of var 0..5: x;
constraint alldifferent(x); % hard constraint
nScs = 1; penalties = [1];
% copied from share/minizinc/std/increasing_int.mzn
predicate my_increasing_int(array[int] of var int: x) =
    forall(i in index_set(x) diff { min(index_set(x)) })
       (x[i-1] <= x[i]);

constraint my_increasing_int(x) <-> satisfied[1];
solve maximize bool2int(satisfied[1]);
output["satisfied[1] = \(satisfied[1]), x = \(x)"];
```

Step 2: What about Globals?

Institute for Software & Systems Engineering

Solved `reified_fix.mzn` with Gecode (same as `reified.mzn` solved with JaCoP or G12-FD):

```
satisfied[1] = false, x = [4, 1, 3, 0, 2]
----------
satisfied[1] = true, x = [0, 1, 2, 3, 4]
----------
==========
```