1 The Nine Children

There is a person with nine children, with equally long gaps between the ages, expressed as integers, of any two consecutive children. The squared age of the parent is the sum of the squares of the ages of the children. Perform the following sequence of tasks:

A. Write a MiniZinc model nineChildren.mzn in order to determine the ages of the children and their parent.

B. Evaluate your model in the report.

C. How old are the children and their parent? Is this solution unique? If not, then give the first solution of each backend.
2  Halmos and his Wife (from Paul Halmos)

Paul Halmos, the mathematician, and his wife attended a dinner party attended by four other couples. During the cocktail hour, some of those present shook hands, but in an unsystematic way, with no attempt to shake everyone’s hand. Nobody shook his or her own hand, nobody shook hands with his or her spouse, and nobody shook hands with the same person more than once. During dinner, Halmos asked each of the nine other people present, including his wife, how many hands they had shaken. Under the given conditions, the possible answers range from 0 to 8. However, it turned out that each person gave a different answer: one person had not shaken any hand, one person had shaken exactly one hand, one person had shaken exactly two hands, and so on, up to one person who had shaken hands with all the others present, except his or her spouse, that is eight hands. Perform the following sequence of tasks:

A. Write a MiniZinc model `wifeHalmos.mzn` in order to determine how many hands Halmos’ wife shook.

B. Evaluate your model in the report, writing the found number of hands instead of ‘sat’ (for ‘satisfiable’) in the performance table.

C. Is the reported solution unique? If not, then report under Task [3] the first solution of each backend.

3  Five Men and a Monkey (from Martin Gardner) (optional)

Five men and a monkey were shipwrecked on a desert island, and they spent the first day gathering coconuts for food, piled them all up, and went to sleep. When they were all asleep, one man woke up and thought there might be a row in the morning about dividing the coconuts, so he decided to take his share. He divided the coconuts into five equal-sized piles. He had one coconut left over, gave it to the monkey, hid his pile, and put the rest back together. The next man woke up and did the same thing; he also had one coconut left over and gave it to the monkey. Eventually, all five men had done the same thing, one after the other, each taking a fifth of the remaining coconuts in the pile when he woke up, and each having one coconut left over for the monkey. In the morning, the monkey had five coconuts and the men divided what coconuts were left: they came out in five equal shares. Of course, each man knew there were coconuts missing, but each man was as guilty as the others, so they did not say anything. Perform the following sequence of tasks:

A. Write a MiniZinc model `fiveMenMonkey.mzn` in order to determine the number of coconuts at the beginning.

B. Evaluate your model in the report, writing the found number of coconuts instead of ‘sat’ (for ‘satisfiable’) in the performance table.

C. Is the reported solution unique? If not, then report under Task [3] the first solution of each backend.

D. What is the minimal solution, and how do you determine it?
4 The Largest Permutation

Find a permutation \([v_1, \ldots, v_n]\) of the integers 1 to \(n\) such that the sum of the products of all pairs of successive values \(v_i\) and \(v_{i+1}\) is maximal. Perform the following sequence of tasks:

A. Write a MiniZinc model `largestPerm.mzn` in order to determine such a permutation.

B. Evaluate your model in the report, from \(n = 5\) to at most \(n = 100\), by steps of 5, using any time-out of 30 to 60 CPU seconds. This problem can be solved in time polynomial in \(n\) but is a difficult benchmark for general-purpose solvers: you are not expected to find a model that scales up to \(n = 100\) before timing out with such a short time-out, and our purpose only is to observe the different scalability of the various solving technologies and backends.

5 Square Packing

Place \(n\) squares of sizes \(1 \times 1, 2 \times 2, \ldots, n \times n\) inside a bounding rectangle of width \(w\) and height \(h\) such that no squares are overlapping and \(w \cdot h\) is minimal. Perform the following sequence of tasks:

A. Write a MiniZinc model `squarePacking.mzn` to determine such a square packing, using the `diffn` predicate (see [http://www.minizinc.org/doc-lib/doc-globals-packing.html](http://www.minizinc.org/doc-lib/doc-globals-packing.html)). Hint: Give the variables as tight domains as possible.

B. Evaluate your model in the report, from \(n = 4\) to \(n = 15\), by steps of 1, using any time-out of 30 to 60 CPU seconds, writing the expression \(\langle w, h, w \cdot h \rangle\) instead of just the objective value \(w \cdot h\) in the performance table.

C. A reflection or rotation of any (partial) solution results in a symmetrical (partial) solution. Likewise, reflecting or rotating any (partial) non-solution results in a symmetrical (partial) non-solution. An easy way to break these symmetries is to constrain the largest square, of size \(n \times n\), to be placed in the lower-left quadrant of the bounding rectangle, that is \(x[n] < w/2\) and \(y[n] < h/2\). Add such a symmetry-breaking constraint to your model, re-run the evaluation of Task [B] and also give the performance impacts this has for each backend, in terms of both solution quality and solution time. Hint: Avoid using the `div` function by rewriting a constraint such as \(a < b \text{ div } c\) into \(a \cdot c < b\).

Trivia: The largest known optimal solution is for \(n = 32\): it has a bounding rectangle of size \(85 \times 135\) and took about 33.5 days to find and prove optimal. Finding this solution requires much more sophisticated symmetry breaking as well as search heuristics.

Submission Instructions

All task answers, other than models, must be in a single PDF report; all other formats are rejected. Furthermore:

- Identify the team members inside the report.
- Certify that your report and all its submitted attachments were produced solely by your team, except where explicitly stated otherwise and clearly referenced, that each teammate can individually explain any part starting from the moment of submitting your report, and that your report and attachments are not and will not be freely accessible on a public repository.
• Address each task of each problem, using the numbering and the ordering in which they appear in the homework statement.

• Take the instructions of the demo report at http://user.it.uu.se/~pierref/courses/M4CO/demoReport as a strict guideline for the structure and content of a model description and evaluation in the report, as well as an indication of its expected quality.

• Write clear task answers, models, and comments.

• Justify all task answers, except where explicitly not required.

• State any assumptions you make that are not in this document.

• Thoroughly proofread, spellcheck, and grammar-check your report.

• Submit all models.

• Match exactly the uppercase, lowercase, and layout conventions of any filenames and I/O texts imposed by the tasks, as we will process your models automatically.

• Write a paragraph, which will not be graded, describing your experience with this assignment: which aspects were too difficult or too easy, which aspects were interesting or boring? This will help us improve the course in the coming years.

Only one of the teammates submits the solution files (one PDF report with answers to all the tasks, and all model files) by email to the assistant (http://www.it.uu.se/katalog/gusbj192), ideally by the suggested deadline.

Grading Rules

If all tasks have been seriously attempted and all requested models exist, have the comments exemplified in the demo report, and produce correct outputs for some of our grading instance data in reasonable time on the Linux computers of the IT department, within the given time bounds (if any), then you get at least 1 point (of 5), otherwise you get 0 points. Furthermore:

• If your models pass most of our grading tests, then you get 4 or 5 points, depending also on the quality of the model comments and report.

• If your models fail many of our grading tests, then you get an initial mark of 1 or 2 points, depending also on the quality of the model comments and report.

However, if the assistant figures out a minor fix that is needed to make your model run as per our instructions above, then, instead of giving 0 points up front, the assistant may deduct 1 point at his discretion.

You must get minimum 1 point (of 5) on this assignment and minimum 8 points (of 15) over all three assignments in order to pass the Assignments part (3 credits) of the course.