Programming Languages Assignments (v1.0)

Eric Laermans (eric.laermans@intec.UGent.be)

2014-03-04

Assignment 1 Lazy execution

This assignment is inspired from two exercises of Chapter 4 of the book "Concepts, Techniques, and Models of Computer Programming": exercises 4.15 and 4.17. The goal is to compare the advantages and drawbacks of lazy evaluation and eager evaluation.

Assignment 1.1 Performance issues

1. Write a few (small) benchmark programs (in Oz): a first version using lazy evaluation, a second version using eager evaluation.

2. To make the difference between the eager and the lazy version clearer, you can replace built-in operations by explicit (lazy) functions. For example addition becomes `Add`, which is defined as `fun {Add X Y} X+Y end` (for the eager version) or as `fun lazy {Add X Y} X+Y end` (for the lazy version).

3. Choose your test programs with care. They must allow you to perform a *quantitative* analysis of the performance impact of lazy evaluation vs. eager evaluation (in Oz). Compare the performances for various input sizes. Make sure the eager and lazy versions of the test programs are comparable.

4. Compare the behaviour of both versions of these programs. What is the difference in efficiency? Can you explain this result? What are the advantages and drawbacks of lazy evaluation?

5. Make sure that your report includes sufficient information to allow a reproduction of the test results.

A few possible choices for benchmark programs are:

- a program implementing the Sieve of Eratosthenes
- a program implementing a list sorting algorithm

All these programs have an input parameter that can easily be adapted (which allows you to tune the total computation time easily).

Assignment 1.2 The Hamming problem

Finally, you will also create a program (in Oz) solving the Hamming problem (see exercise 4.17), again in a lazy version and in an eager version.

1. Here, the eager and lazy version needn't be totally comparable (why?).
2. Make sure the eager version is sufficiently efficient.
3. Compare the performances of both versions as a function of the input parameters.
4. Which version is to be preferred? Why?

Assignment 2 Declarative model and concurrency

Assignment 2.1 Declarative model

You probably know the principle of sudokus. The "sum sudoku" (aka "killer sudoku") is a variant, where given areas are outlined by dashed lines (aka "cages") and each of these cages mentions the sum of all numbers within that cage. The other rules from traditional sudoku still apply: the integers ranging from 1 to 9 are used; no integer occurs more than once in the same column, row, or bordered 3-by-3 square. An example is given in Fig. 1 (taken from the 2008 World Sudoku Championship).

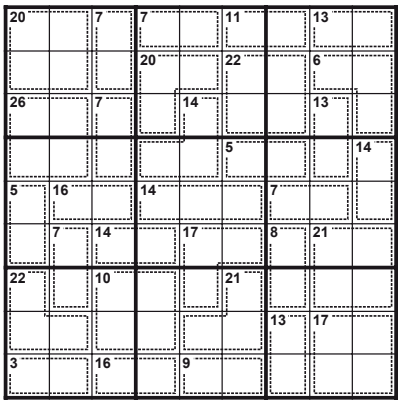

Figure 1: A 3-by-3 killer sudoku

We say that a sudoku is *solved* iff none of its cells are empty. All and only solutions of a sudoku S are solved sudokus obtained by adding numbers to the empty cells of S (always respecting the rule that no integer occurs more than once in the same column, row, or square, and —for a killer sudoku— that the sum condition for each cage is fulfilled). To solve a sudoku is to find all its solutions. A sudoku is *sound* iff it can be solved. A sudoku is *complete* iff it has only one solution. Most published sudokus are both sound and complete (as is the one shown in Fig. 1). You may assume for this assignment that only sound and complete killer sudokus are considered.

1. Write a program in the declarative model [1] that, when given a killer sudoku, returns its solution (you may assume the solution is unique).

2. Use it to solve the sudoku shown in Fig. 1. Don't forget to mention both result and required computation time.

Assignment 2.2 Introducing Concurrency

A variation on 3-by-3 killer sudokus are linked killer sudokus. An example is shown in Fig. 2 (still from the 2008 World Sudoku Championship).

[1] It isn't mandatory to use Oz for that purpose, but if you choose to program in a (more familiar) "stateful language" (e.g. Java, C or C++), explain how you can do this in a "declarative way". If you don't use Oz, please also add an executable version of your program, as this would save me compiling troubles.

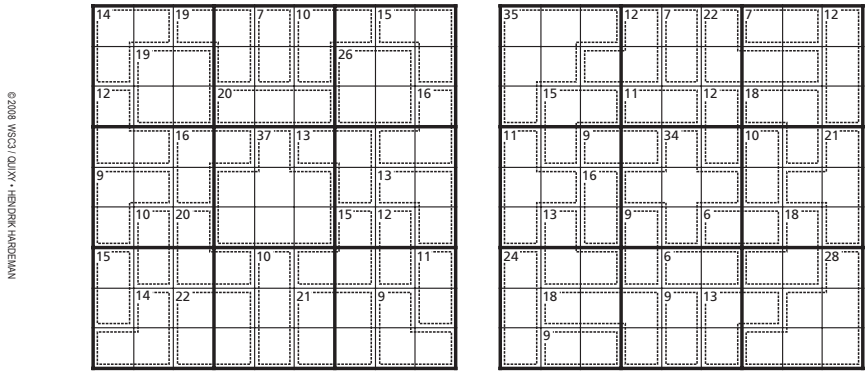

Figure 2: Linked killer sudokus

They can be seen as 2 distinct killer sudoku puzzles, with different sum clues and different cages, but with identical solution. It is generally not possible to fully solve the individual killer sudokus separately. The sum clues from both killer sudokus are necessary to find the unique solution of the linked killer sudoku.

1. Adapt your program for solving 3-by-3 killer sudokus into a program that concurrently solves the 2 killer sudoku puzzles, each in a separate thread. The threads will need some communication mechanism to solve the linked killer sudoku.

2. Choose one of the different possible concurrent models (declarative concurrent, message-passing concurrent, stateful concurrent,...) and explain your choice. What would have been the advantages and drawbacks of the other concurrent models?

3. Use the concurrent program to solve the linked sudoku shown in Fig. 2. Don't forget to mention both result and required computation time.

Practically

Report

Consider your report as a paper describing the work you have done. So, don't forget to mention the hardware/software on which the programs are executed and don't forget a conclusion for each assignment. Also be honest about possible remaining issues with the code you have written. I prefer to read about it in the report than to discover it when I test the code.

It would be greatly appreciated if you could add some explanation about your code in your report, as I'd rather not delve into programming code to try and understand how your code works.

The assignments (software + report) are due on **May 19, 2014** and should be done individually. The assignments (including the discussion of the results at the oral exam) will count for 35% of your global score.

As the first exams are scheduled on May 26, it is impossible for me to give feedback about the score of your assignments before the exam. If you want earlier feedback, please ask this in advance. However, in this case, I expect your report on May 9, 2014 (10 days before the normal deadline).

Questions

If you have any further question, please contact me.