Programming Language Design
Assignment 3 2021

Torben Mogensen and Hans Hüttel

9th March 2021

This assignment is *individual*, so you are not allowed to discuss it with other students. All questions should be addressed to teachers and TAs. If you use material from the Internet or books, cite the sources. Plagiarism *will* be reported.

Assignment 3 counts 25% of the grade for the course, but you are required to get at least 33% of the possible score for every assignment, so you can not count on passing by the later assignments only. You are expected to use around 20 hours in total on this assignment, including the resubmission (if any).

The deadline for the assignment is **Friday March 19 at 16:00 (4:00 PM)**. In normal circumstances, feedback will be given by your TA no later than March 29. The individual exercises below are given percentages that provide a rough idea how much they count (and how much time you are expected to use on them). These percentages do *not* translate directly to a grade, but will give a rough idea of how much each exercise counts.

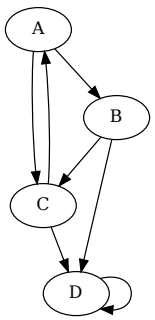
We strongly recommend you to resubmit your answer after you have used the feedback to improve it. You can do so until **April 6 at 16:00 (4:00 PM)**. Note that resubmission is made as a separate mandatory assignment on Absalon. If you resubmit, the resubmission is used for grading, otherwise your first submission will be used for grading.

The assignments consists of several exercises, some from the notes and some specified in the text below. You should hand in a single PDF file with your answers and a zip-file with your code. Hand-in is through Absalon. The assignment must be written in English.

A3.1) (30%) Comparing Graphviz to LaTeX graph tools.

Section 11.5.3 in the notes present Graphviz (`https://www.graphviz.org/`), a DSL for drawing graphs and trees, and shows two different graphs produced by Graphviz. The first was produced by the command `dot example.gv -Tpdf > example.pdf`, where `example.gv` has the Graphviz code shown in the notes, and `example.pdf` is a PDF file with the resulting graph. The option `-Tpdf` specifies that the output format is PDF.

a. Install Graphviz. Use `sudo apt install graphviz` or follow the instructions on `https://www.graphviz.org/` and reproduce the following graph using Graphviz:



Show the Graphviz code in your report. You should be able to use the example in the notes as a guide without having to read the Graphviz documentation.

b. Reproduce the same graph in LaTeX, using your favourite graphics package (such as PSTricks, TikZ, or just the picture environment). The layout does not have to be the same, as long as

the same nodes and edges are represented. For example, you can use rectangular boxes instead of ellipses, straight arrows instead of curved, and you can change the placement of nodes.

c. Discuss the effort of making such graph drawings using Graphviz compared to using a graphing package in LaTeX.

d. As shown in the notes, when the graph is modified by adding a single edge, Graphviz can drastically change the layout of the drawing. Consider advantages and disadvantages of this behaviour.

A3.2) (15 %) Here is a function written in SML:

```
fun dingo (x, nil) = false
  | dingo (x, y::ys) = x=y orelse dingo (x,ys);
```

(a) Explain what this function does.

(b) Use the Hindley-Milner algorithm to infer the type of dingo; describe the steps that you carry out.

A3.3) (20 %) In the session on logic programming we saw the following definition of lists.

```
list(nil).
list(cons(A,As)):- list(As).
```

Extend this program with a predicate longerthan in Prolog such that longerthan(L1,L2) holds if the list L1 is longer than the list L2.

A3.4) (20 %) Let us add the following new loop construct to the statements in the imperative language presented in the session on program semantics.

$$\texttt{repeat}\ s\ \texttt{until}\ x$$

The intention behind the new language construct is that the body s is executed until the value of x is 0. This implies that s is always executed at least once.

Extend the *operational semantics* of the imperative language with transition rules for the new loop construct and explain what happens in the rules.

A3.5) (15 %) Let us add the following construct to the functional language presented in the session on program semantics.

$$\texttt{private}\ x = e_1\ \texttt{within}\ e_2$$

The intention is that we inside e_2 (but only there) can mention the variable x and that its value is that of the expression e_2. For instance, the expression

$$\texttt{private}\ x = 5\ \texttt{within}\ x + 12$$

should evaluate to the value 17.

Extend the *type rules* of the functional language with rules for this new construct and explain what happens in the rules.