Programming Language Design 2022
Functional Programming and Control Structures

Hans Hüttel

22 February 2022

Where nothing else is mentioned, chapters and exercises are from the notes "Programming Language Design and Implementation", which are available on the Absalon page for the course.

The slides from the podcast can be fouwnd in the "Slides" folder, with LaTeXsources in the "Slide sources" folder. You can cut-and-paste from these to your assignment reports.

Many of the exercises has no single correct answer, so you are expected to prepare for discussing these in the exercise classes.

The video podcast and the slides

The video podcast is available from the main course page on Absalon. You can either watch the podcasts via the media library or download everything. The podcast is yours to keep. All files, podcasts and slides alike, have the same name as the topic of the session. Remember that the podcasts come in several parts.

If you prefer to watch the podcasts online, you can find them in the Media section and watch them there. If you want to download the podcast and watch everything off-line, visit the Video folder in the Files area. The corresponding slides can be found in the Slides folder along with their LaTeX sources in the Slide sources' folder. You can cut and paste from these to your assignment reports.

Many of the exercises have no single correct answer, so you are expected to prepare for discussing these in the exercise classes.

Tuesday 22 February – *Functional Programming and Control Structures*

The podcast and slides will cover Section 6.5 and Chapter 7 of "Programming Language Design and Implementation".

Learning goals for the session *F*unctional Programming and Control Structures

The learning goals for this session are

Learning goals – Functional languages

- To be able to give simple examples of higher-order functions and explain why they are higher-order
- To be able to explain the notion of closures and why they are important for static scope rules
- To be able to explain the principles of the program transformations that make it possible to implement higher-order functions when we assume static scope rules: λ-lifting, closure conversion and defunctionalization.
- To be able to apply these program transformations.

Learning goals – Control structures

- To be able to classify control structures in imperative and functional programming languages.
- To be able to explain the various notions of jumps and their pros and cons.
- To be able to explain the notion of list homomorphism and why it is useful.
- To be able to explain why jumps can be implemented in a language without jumps.
- To be able to explain the characteristics of the various forms of multithreading.

How you should prepare before we meet on 22 February

Before we meet on 21 February in Aud. 5 at HCØ, watch the podcast (all parts, please!) and read the text. You can do this in any order you like.

Also see if you can come up with answers to the following small problems.

1. In an earlier version of the slides, the author wrote that we can think of a closure as a closed curried function of two arguments that has been applied to first argument, e.g. `f 5`. Why does this make sense?
2. In the podcast it is claimed that scope rules for exception handlers should by dynamic. But why would one want dynamic scope rules here?

What happens on 21 February?

When we meet, we will discuss the small problems that I mentioned above along with a bunch of other small problems that I will make available to you on the day. The goal of this is to help you understand the text better and to prepare you for the exercise session on Tuesday 22 February.

I will introduce the small problems along the way and ask you to discuss them in small groups of two or three people. Every now and then I will discuss the problems with everyone.

Along the way, there will of course also be time for you to ask me any questions that you may have. I will do my best to answer them.

Tuesday 22 February – *Exercises*

For the exercise class (13:00-15:00), prepare the following exercises for presentation and discussion in (online) class:

1. In the podcast we did not specify what should happen if a coroutine executes to the end of its body. What are some possibilities, and what are the pros and cons?
2. Below is a Haskell program P.

```
f x = let dingo z = z + x
    in
        let
           bingo w = dingo (47-w)
        in
           g (bingo 419)

g u = let mango x = x + h u
    in
        let
           dingo k = k
        in
           mango (dingo u)

h w = 42
```

   - What is `f 14`?
   - Use the λ-lifting algorithm to transform P.

3. Is there any need for *selective* control structures whose tests are evaluated sequentially, such as the following?

```
if i > n orelse a[i] = x then ... else ...
```

4. In ALGOL68 there is only one loop construct, whose most elaborate form is

   for I **from** E1 **by** E2 **to** E3 **while** E4 **do** C **od**

   where E1, E2, E3 and E4 are expressions that are integer-valued and are evaluated only once. The scope of the identifier I is E4 and C.

   The only part that cannot be omitted is C.

   - How many different control structures can we express using this general loop construct?

- What do you expect will happen if we omit the following?
  a) **for** I
  b) **from** E1
  c) **by** E2
  d) **to** E3
  e) **while** E4
- Can we express the repeat-until loop construct using this general loop only? If yes, describe how this is done. If no, explain the difficulties and if there is a solution.
- How might this general loop construct be extended such that we could use it as an *expression*?