LVA 185.A05 Advanced Functional Programming (SS 2020)

Self-Assessment Test 8

Friday, 29 May 2020

Topics: All Parts, All Chapters

Aspects of Advanced Functional Programming

(No submission, no grading)

All Parts, All Chapters

– What is

1. monadic programming (dealt w/ e.g. in Chapter(s) ... on ...)
2. stream programming (dealt w/ e.g. in Chapter(s) ... on ...)
3. abstract data type programming (dealt w/ e.g. in Chapter(s) ... on ...)
4. algorithm pattern programming (dealt w/ e.g. in Chapter(s) ... on ...)
5. generate/prune programming (dealt w/ e.g. in Chapter(s) ... on ...)
6. memoization programming (dealt w/ e.g. in Chapter(s) ... on ...)
7. 'Münchhausen-style' programming (dealt w/ e.g. in Chapter(s) ... on ...)
8. combinator programming (dealt w/ e.g. in Chapter(s) ... on ...)
9. (functional) 'glue' programming (dealt w/ e.g. in Chapter(s) ... on ...)
10. functional array programming (dealt w/ e.g. in Chapter(s) ... on ...)
11. functional pearls programming (dealt w/ e.g. in Chapter(s) ... on ...)
12. functional reactive programming (dealt w/ e.g. in Chapter(s) ... on ...)
13. higher-order functions programming (dealt w/ e.g. in Chapter(s) ... on ...)
14. input/output programming (dealt w/ e.g. in Chapter(s) ... on ...)
15. functional logic programming (dealt w/ e.g. in Chapter(s) ... on ...)
16. parallel functional programming (dealt w/ e.g. in Chapter(s) ... on ...)
17. lawful programming (dealt w/ e.g. in Chapter(s) ... on ...)
18. dynamic programming (dealt w/ e.g. in Chapter(s) ... on ...)
20. corecursive programming (dealt w/ e.g. in Chapter(s) ... on ...)
20. functional programming (dealt w/ e.g. in Chapter(s) ... on ...)
21. equational reasoning (dealt w/ e.g. in Chapter(s) ... on ...)
22. program calculation (dealt w/ e.g. in Chapter(s) ... on ...)

Use also examples to explain and illustrate the essence, the benefits but also challenges related to the above notions.

– What are important approaches or techniques, means at the disposal for a functional programmer to gain confidence in or ensure correctness of a functional program on a

23. coarse-grained
24. fine-grained

level? Is there specific tool or language support in Haskell/other functional programming languages to support functional programmers following or applying these approaches?

25. Sketch the historical evolution of programming paradigms and languages. What was the driving ambition, goal and purpose pushing this evolution so far?

26. Every programming paradigm and every programming language can make it to become the main stream paradigm and language. In principle, at least. Do you agree? Name and discuss technical and non-technical barriers which can make this more or less easy in reality.

27. What are particular strengths of the functional programming paradigm compared to the object-oriented one? What are particular strengths of the object-oriented programming paradigm compared to the functional one?

28. There is no consensus whether functional languages should be eagerly or lazily evaluated. What is your view on this question? Why? What are arguments in favour of lazy and eager evaluation, respectively?

29. Functional programming and (embedded) domain-specific languages seem to go well along with each other. What are reasons for this? What are examples of domain-specific languages in this course?

30. The term 'functional imperative programming' seems contradictory in itself. Do you agree? What is meant?

31. Explain and illustrate the idea of a parallel map.

32. I have experiences with several programming paradigms and languages. Based on these experiences

    - my favorite programming style is ... because ...
    - I do not have a favorite programming style because ...

33. Based on my experience, the below programming task should best be solved using a/an ... programming style because ...

34. The Thue-Morse sequence is a stream of `0`'s and `1`'s that is produced as follows: First produce `0`. Next, at any stage, swap everything that was produced so far (by interchanging `0`'s and `1`'s) and append that.

    The first few stages of producing the sequence looks like this:

    ```
    0
    01
    0110
    01101001
    0110100110010110
    ```

    Thus, if A_k denotes the first 2^k symbols of the sequence, then A_{k+1} equals $A_k + +B_k$, where B_k is obtained from A_k by interchanging `0`'s and `1`'s.

    Can you give a corecursive program producing the Thue-Morse sequence as a stream?

35. Topics most challenging in functional programming

    - in general
    - in this course

    I consider ... because ...

36. A subject related to functional programming I would like to dive into in more detail and depth is ... because ...

37. For my independent studies I did refer frequently, sometimes, occasionally, by need/did not refer to

    - textbooks
    - journal articles
    - conference papers
    - conference presentations (e.g., slides, videos of presentations, where available)
    - Web resources (e.g., `haskell.org`)
    - other resources

because ...

38. For me, most useful supporting my independent studies were ... because ...
39. Based on my experience, distance learning is
    - a perfect/inadequate/'depends' substitute/complement of face-to-face teaching
    - superior/inferior to face-to-face teaching
    - a teaching/learning style of its own, neither superior nor inferior to face-to-face teaching, just different with its own strengths, limitations, and short-comings
    - perfect/inappropriate/'depends' as the sole mode of teaching/learning for me/for me for particular topics

    because ...
40. If I had a choice I would go for
    - distance teaching and learning
    - face-to-face teaching and learning
    - mixed/blended forms of distance and face-to-face teaching and learning
    - topic-dependently for one of the above

    in the future because ...
41. Are you planning to take part in this year's programming challenge at ICFP 2020 (cf. Chapter 20)? Did you already form a team?
42. What question(s) related to functional programming you are (still) puzzled by?