MAT605 Logic and Foundations with Haskell

Marius Furter

March 17, 2023

1 Resources

The video material for the course may be found either on SWITCHcast or on YouTube. In both places the videos contain chapter headings to help you navigate the material.

2 Learning goals

The following is a summary of the learning goals of the course by week. Each learning goal for the theory will be marked with one of the following symbols:

🙂 **(essentials)** These points are essential knowledge for passing the course. If you complete all of them, you should have no trouble passing the exam.

🤔 **(food for thought)** These points are more advanced and might require more work to understand. They are not required in order to get a passing grade, but some of these topics might appear on the exam.

🤯 **(exploding head)** These points are completely optional. They are either difficult, tedious, or things I included for your culture. They will not appear on the exam, but they might help you understand the other material better.

Aside from the learning goals, I will also provide some questions for you to contemplate.

Week 1

Haskell 1 :: Setup This video covers how to get Haskell installed on your computer. I recommend using GHC together with Visual Studio Code. In order to make the two work together, you need to let GHCup install all the stuff it asks you to, in particular you need the Haskell language server. In VSCode you will need to install the `Haskell` and `Haskell Syntax Highlighting` extensions.

Course Intro This video introduces the course. Watching it is optional.

Logic 1 :: Introduction This video introduces the logic portion of the course. It doesn't contain any essential material, but it sets up some ideas that will be helpful later. I would recommend watching it.

Question: Think about some mathematical statements of your favorite area. What does it mean for such statements to be 'true'? How would you translate this to formulas in a logical system?

Logic 2 :: Naive Propositional Logic This video covers the basics of propositional logic in an informal manner. We will see all of this in more detail in a few weeks. You probably have already encountered all of this material in your studies, but I included it so as to not assume any prior knowledge. If you feel like you satisfy the learning goals, feel free to skip this video. If you are unclear on some points, I would recommend skimming the video using the chapter feature.

- 🙂 Understand the *syntax* of propositional logic. Be able to recognize and write well-formed formulas.
- 🙂 Know the *truth table definitions* for the usual logical connectives ($\neg$, $\wedge$, $\vee$, $\rightarrow$, $\leftrightarrow$). Be able to calculate the truth value of propositional logc formulas.
- 🙂 Know the definition of *logical validity* and *logical equivalence* for propositional formulas. Be able to demonstrate these using truth tables.
- 🙂 Be able to prove new logical equivalences from old ones by *substitution*.

Week 2

Haskell 2 :: Basic Operations This video covers basic operations in Haskell such as arithmetic, comparison, lists, ranges, list comprehension, and simple functions. It follows chapter 2 of the "Learn you a Haskell for great good" tutorial.

- 🙂 Be able to do *basic arithmetic* in Haskell
- 🙂 Be able to *compare* objects
- 🙂 Be able to define *lists and ranges* and understand their limitations
- 🙂 Understand the syntax for *list comprehension*
- 🙂 Know how to write *basic functions*
- 🙂 Know the difference between *prefix and infix functions*. Be able to convert each into the other
- 🙂 Be able to define *tuples* and understand their limitations

Haskell 3 :: Types and Typeclasses This video covers the type system in Haskell. It follows chapter 3 of the "Learn you a Haskell for great good" tutorial.

- Know the *basic types* listed in the online tutorial. Be able to create objects of each of them.
- Know the syntax for declaring the *types of function* with one or more arguments
- Know how to use *type variables and typeclasses* in type declarations for functions
- Be familiar with the most common *typeclasses* (Eq, Ord, Num, Integral, Floating)

Logic 3 :: Naive First Order Logic This video covers first order logic informally based on examples. You probably already are familiar with much of the material in this video. I would recommend watching it if you haven't thought a lot about the structure of first order logic formulas before. You will then be better prepared for the formal definitions that will come later.

- Be able to parse well-formed first order logic formulas into a *parsing tree*
- Be able to *translate* between precise mathematical statements and first order logic formulas
- Be able to convert between *restricted quantifiers* and unrestricted ones

Week 3

Haskell 4 :: Functions This video covers how to define functions using pattern matching and guards. It also discusses `where`, `let`, and `case`.

- Know the *syntax for pattern matching* numbers, tuples, and lists.
- Know the *syntax* for defining functions using *guards*.
- Know the *syntax* for `where` and `let`.

Logic 4 :: Informal Proof Theory This video covers all the rules necessary to prove first order logic formulas. The main goal is for you to know and be able to apply these rules. Since most of you will already have quite some experience in proving things, this material should seem familiar. However, you might not have explicitly given the rules much thought. For additional examples and exercises, I recommend looking at Velleman's book "How to prove it".

- Know the proof rules for all first order logic connectives. I have summarized the rules here.

- 🤔 Be able to *write basic proofs* using the rules.
- 🙂 Be able to recognize the *logical equivalences* involving quantifiers presented at the end of the video.

Week 4

Haskell 5 :: Implementing Logical Functions In this video we implement our own version of the type Bool along with functions acting on Bool.

- 🙂 Be able to write functions on the level of the translation functions and the logical operators `not`, `&&`, ||.
- 🙂 Be able to understand the recursive definitions for `and`, `or`, `elem`, `all`, `any`, and `filter`.
- 🤔 Be able to implement functions on the level of `and`, `or` using recursion.