04Lab
Verified Specification, ADTs, type classes, and monads

Mirko Viroli
mirko.viroli@unibo.it

C.D.L. Magistrale in Ingegneria e Scienze Informatiche
Alma Mater Studiorum—Università di Bologna, Cesena

a.a. 2023/2024

One slide sum-up on Advanced Testing

Software models and programming languages

- software models are specification of a software systems, useful for design and verification
- software models capturing system behaviour are systems per se, to be properly engineered
- programming languages (like Scala) are very good at programming software models

An example formal model: ADTs

- type, constructors, operators, and axioms (or other math)
- can be turned into Scala ADTs, with trait/opaque types, and using ScalaCheck/Test

From ADTs to type-classes and monads

- type-classes can be used to factorise empowerment of existing, simple ADTs
- an example is monads, a flexible way to compose functional abstractions
- among the many applications, one allows you to write an entire MVC application

Starting point and goals

References

- 04-repo: `https://github.com/mviroli/course-asmd23-04-adv-prg-patterns`
- ScalaTest: `https://www.scalatest.org/user_guide`
- ScalaCheck: `https://scalacheck.org/documentation.html`
- ADT: `https://softwarefoundations.cis.upenn.edu/vfa-current/ADT.html`

General goals

- be operative with ScalaCheck/Test
- play with ADTs and ScalaCheck
- play with monads
- engineer with monads

Tasks

TEST-OPERATE

Download the repo and check everything works as expected. Play just a bit with ScalaCheck and see which parameters it has (e.g. number of generated tests?). Play also with ScalaTest, and see if it can perform parameterized tests. What are key differences between the two?

ADT-VERIFIER

Define an ADT for sequences with some operations: `map/filer/flatMap/fold/reduce/...`. Turn into a Scala trait, and define ScalaCheck properties capturing axioms 1-to-1. Develop two implementations (Cons/Nil and by Scala List). Engineer tests such that you can easily show they both satisfy those properties.

MONAD-VERIFIER

Define ScalaCheck properties for Monad axioms, and prove that some of the monads given during the lesson actually satisfy them. Derive a general methodology to structure those tests.

MVC-ENGINEER

Start with the given MVC monadic application: extend it to realise a more complex application, e.g. the DrawNumberGame. Of course, be fully monadic. Up to which complexity can one reach? Could we come up with a simple MVC application with reactive GUI, and/or could a GameLoop be framed into a fully monadic application?

ADVANCED-FP-LLM

LLMs/ChatGPT can arguably help in write/improve/complete/implement/reverse-engineer ADT specifications, ADTs in Scala, and monadic specifications. Check if/whether this is the case.