Chapter 10

Conclusions: Overall structure

Overall structure

1. Introduction
2. Functional specification and programming
3. Language and semantical aspects of higher-order logic
4. Proof system for higher-order logic
5. Sets, functions, relations, and fixpoints
6. Verifying functions
7. Inductively defined sets
8. Specification of programming language semantics
9. Program verification and programming logic

Chapter 1: Introduction

1. Give an overview of the course.
2. Explain the terms model, specification, verification.
3. Explain language and semantics of propositional logic.
4. Give and explain a logical rule. How is this rule applied?
5. What is a Hilbert style, what a natural deduction style proof system?
6. What is the advantage of a Hilbert style proof system?
7. Why is a natural deduction style proof system chosen for interactive proof assistants?

Chapter 2: Functional programming and specification

1. What is the relationship between the data type construct and the case expression? Illustrate the relationship by an example.
2. What is the meaning of “fun f x = f x” in ML, what is the meaning of the corresponding definition in Isabelle/HOL?
3. Why are there different forms of function definitions in Isabelle/HOL, but only one in ML?
4. Why is there a distinction between types with equality and types without equality in ML, but not in Isabelle/HOL?

Chapter 3: Language and semantical aspects of HOL

1. What is the foundational reason that HOL is typed? Are there other reasons w.r.t. an application in computer science?
2. What does “higher-order” mean?
3. Why is predicate logic not sufficient? Give an example?
4. What are the types in HOL?
5. What are the terms in HOL? Give examples of constants.
6. Explain the description operator.
7. What is a frame? What is an interpretation?
8. How is satisfiability defined?

9. What is a standard model?
10. Give and explain one of the axioms of HOL?
11. Can the constants True and False be defined in HOL?
12. What does it mean that HOL+infinity is incomplete wrt. standard models?
13. What is a conservative extension?
14. What is the advantage of conservative extensions over axiomatic definitions?
15. Which syntactic schemata for conservative extensions were treated in the lecture?
16. Give examples of constant definitions.
17. Explain the definitions of new types?
18. Does a data type definition in Isabelle/HOL lead to a new type?

Chapter 4: Proof system for HOL

1. A natural deduction proof system distinguishes between formulas, sequents, and rules. What are the differences?
2. Isabelle/HOL has nor clear distinction between sequents and rules. Why?
3. Explain the different kinds of variables.
4. What is a proof state?
5. What is the distinction between a rule and a method?
6. Explain the method “rule” and show in detail how it can be applied in a proof state?
7. What is an elimination rule?
8. Here is a proof state (shown on the screen). What is a rule that can be applied?

9. Name some rule and their uses.
10. What does it mean that a rule is safe?
11. Why is verification in Isabelle/HOL usually based on theory Main and not directly on the HOL axioms?
12. What is rewriting and simplification?
13. How can an Isabelle/HOL user influence the simplification process?
14. What is case analysis?
15. How differ methods for proof automation?
16. Explain a method for proof automation.
17. What is a forward proof step?

Chapter 5: Sets, functions, relations, and fixpoints

1. What is the relationship between sets and functions?
2. What is set comprehension?
3. How are sets be realized in Isabelle/HOL?
4. Whare is the relationship between sets and types (in Isabelle/HOL)?
5. What is the principle of extensionality for functions? Why is it important for verification?
6. Define injectivity as a predicate in Isabelle/HOL.
7. How are relations represented in Isabelle/HOL. What would be a different representation?

8. How can the reflexive and transitive closure of a relation be defined? Can this be done in first order logic?
9. What is a well-founded relation?
10. What is a measure function?
11. Explain an application of well-founded relations?
12. What is a complete lattice? Give an example of a complete lattice.
13. Explain the Kaster/Tarski theorem. Why is it important? What is the relationship to inductive definitions?

Chapter 6: Verifying functions

1. Explain the difference between verification and testing.
2. What is the advantage of formal proofs over paper and pencil proofs?
3. Property specifications can be wrong. Does this mean that verification is useless?
4. What is the relationship between termination and well-definedness of functions?
5. How is termination usually proved? Sketch this for gcd and quicksort.
6. What are the properties we proved for quicksort?

7. Explain shallow embedding.
8. How can functional properties of algorithms are proven in Isabelle/HOL?
9. Can Isabelle/HOL be used to prove the complexity of an algorithm? What would be needed (together with Chapter 8)?
10. What does structural induction over the function parameters mean?

Chapter 7: Inductively defined sets

1. Explain the inductive definition of sets. What is the syntactic schema used?
2. Why is it necessary to constrain inductive definition to the syntactic schema?
3. Give an example of an inductive definition.
4. What is the relationsship between recursive and inductive definitions?
5. What is a coinductive definition?

6. For which situation are coinductive definitions needed?
7. What is a transition system? Give examples.
8. Explain the syntax of LTL defined in the lecture.
9. What is a Kripke structure? How is it related to transition systems?
10. What is a liveness property?

Chapter 8:
Specification of programming language semantics

1. What is a programming language semantics? Who is a typical user of a semantics?
2. What is a deep embedding of a language into a specification framework such as Isabelle/HOL?
3. Explain big step semantics.
4. What can be expressed in small step semantics that is not directly expressable in big step semantics?

5. Show how the semantics of parallel statement execution can be handled in small step semantics.
6. What does compositionality mean in the context of denotational semantics?
7. How is operational semantics formalized in Isabelle/HOL? Explain motivations for such formalizations.
8. Can programming language semantics be used for program verification?

Chapter 9:
Program verification and programming logic

1. What does it mean that a Hoare triple is valid? How can validity be formalized?
2. How can a programming logic be expressed in HOL?
3. Why are assertions in Hoare logic be formalized as functions?
4. Can Hoare logic proofs be done in Isabelle/HOL? Explain a rule application?
5. What does soundness mean for a Hoare logic? How is soundness proved?