Chapter 9

Questions for Chapters 5 - 8

Overview of Chapter

9. Questions for Chapters 5 - 8

Chapter 5: 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?

Chapter 5: Verifying functions (2)

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 6: Inductive definitions and fixed points

1. What is the relationship between sets and functions?
2. What is set comprehension?
3. How are sets be realized in Isabelle/HOL?
4. What 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?

Chapter 6: Inductive definitions and fixed points (2)

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: Inductive definitions and fixed points (3)

14. Explain the inductive definition of sets. What is the syntactic schema used?
15. Why is it necessary to constrain inductive definition to the syntactic schema?
16. Give an example of an inductive definition.
17. What is the relationsship between recursive and inductive definitions?
18. What is a coinductive definition? framebreak
19. For which situation are coinductive definitions needed?
20. What is a transition system? Give examples.
21. Explain the syntax of LTL defined in the lecture.
22. What is a Kripke structure? How is it related to transition systems?
23. What is a liveness property?

Chapter 7: 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?

Chapter 7: Programming language semantics (2)

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

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?
6. What is a WP-transformer? How is WP-transformation defined for IMP?
7. What do we mean by Hoare logic in WP-form? What is the advantage of Hoare logic in WP-form over the classical rules?
8. How are arrays handled in Hoare logic?

Chapter 9: Program verification (2)

9. How can an abort statement be represented by the abstract Do-command that was considered in the lecture?
10. Explain the while rule in the Hoare logic for total correctness.
11. Why are assumptions needed to verify procedures?
12. What are adaptation rules? What are they used for?
13. What is an abstraction predicate over heaps? Give an example.
14. Explain differences in the heap/store formalizations of SIMPL and Java-KE?
15. Why are triples with virtual methods needed to verify OO programs?
16. What kind of properties should be specified for methods in OO programs?
17. What is extended static checking of programs?