Verständnisfragen

Christoph Lüth

Praktische Informatik 3, WS 20/21

Im folgenden eine Liste von *Verständnisfragen*. Dieses ist *keine* vollständige Liste möglicher Prüfungsfragen; vielmehr sollen diese Fragen das Verständnis des in der Vorlesung vorgestellten Stoffs überprüfen. Wer diese Fragen sicher beantworten kann, ist für eine mündliche Prüfung gut gerüstet; wer sich bei vielen unsicher ist, sollte seinen Tutor oder den Veranstalter kontaktieren.

1. Was bedeutet Striktheit, und welche in Haskell definierbaren Funktionen haben diese Eigenschaft?
2. Was ist ein algebraischer Datentyp, und was ist ein Konstruktor?
3. Was sind die drei Eigenschaften, welche die Konstruktoren eines algebraischen Datentyps auszeichnen, was ermöglichen sie und warum?
4. Welche zusätzliche Mächtigkeit wird durch Rekursion bei algebraischen Datentypen in der Modellierung erreicht? Was läßt sich mit rekursiven Datentypen modellieren, was sich durch nicht-rekursive Datentypen (wie in der Sprache C) nicht erreichen läßt?
5. Was ist der Unterschied zwischen Bäumen und Graphen, in Haskell modelliert?
6. Was sind die wesentlichen Gemeinsamkeiten, und was sind die wesentlichen Unterschiede zwischen algebraischen Datentypen in Haskell, und Objekten in Java?
7. Was ist Polymorphie?
8. Welche zwei Arten der Polymorphie haben wir kennengelernt, und wie unterschieden sie sich?
9. Was ist der Unterschied zwischen Polymorphie in Haskell, und Polymorphie in Java?
10. Was kennzeichnet einfach rekursive Funktionen, wie wir sie in der Vorlesung kennengelernt haben, und wie sind sie durch die Funktion `foldr` darstellbar?
11. Welche anderen geläufigen Funktionen höherer Ordnung haben wir kennengelernt?
12. Gibt es die Funktion `map` auch für andere Typen?
13. Was ist η-Kontraktion, und warum ist es zulässig?
14. Wann verwendet man `foldr`, wann `foldl`, und unter welchen Bedingungen ist das Ergebnis das gleiche?
15. `foldr` ist die "kanonische einfach rekursive Funktion" (Vorlesung). Was bedeutet das, und warum ist das so? Für welche Datentypen gilt das?
16. Wann kann `foldr f a xs` auch für ein zyklisches Argument `xs` (bspw. eine zyklische Liste) terminieren?
17. Warum sind endrekursive Funktionen im allgemeinen schneller als nicht-endrekursive Funktionen?
18. Warum kann ich die Funktion `seq` nicht in Haskell definieren?
19. Welches sind die charakteristischen Eigenschaften von Haskells Typsystem (Hindley-Milner)?
20. Warum ist es hilfreich, Typen abzuleiten und nicht nur die gegebene Typisierung zu überprüfen?

21. Was ist ein abstrakter Datentyp (ADT)?
22. Was sind Unterschiede und Gemeinsamkeiten zwischen ADTs und Objekten, wie wir sie aus Sprachen wie Java kennen?
23. Wozu dienen Module in Haskell?
24. Wie können wir die Typen und Operationen der Signatur eines abstrakten Datentypen grob klassifizieren, und welche Auswirkungen hat diese Klassifikation auf die zu formulierenden Eigenschaften?
25. Warum "finden Tests Fehler", aber "zeigen Beweise Korrektheit", wie in der Vorlesung behauptet? Stimmt das immer?
26. Müssen Axiome immer ausführbar sein? Welche Axiome wären nicht ausführbar?
27. Was kennzeichnet Aktionen?
28. Warum sind Aktionen nicht explizit als Zustandsübergang modelliert, sondern implizit als abstrakter Datentyp `IO`?
29. Warum ist die Erzeugung von Zufallszahlen eine Aktion?
30. Was ist (außer dem Mangel an referentieller Transparenz) die entscheidende Eigenschaft, die Aktionen von reinen Funktionen unterscheidet?
31. Was ist die Typklasse `Functor` ?
32. Wozu dienen Monaden in Haskell?