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?

1. Welche zusätzliche Mächtigkeit wird durch Rekursion bei algebraischen Datentypen in der Modellierung erreicht? Was läßt sich mit rekursiven Datentypen modellieren, was sich nicht durch nicht-rekursive Datentypen erreichen läßt?

2. Was ist der Unterschied zwischen Bäumen und Graphen, in Haskell modelliert?

3. Was sind die wesentlichen Gemeinsamkeiten, und was sind die wesentlichen

Unterschiede zwischen algebraischen Datentypen in Haskell, und Objekten in Java?

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?

1. Was kennzeichnet strukturell rekursive Funktionen, wie wir sie in der Vorlesung kennengelernt haben, und wie sind sie durch die Funktion foldr darstellbar?

2. Welche anderen geläufigen Funktionen höherer Ordnung kennen wir?

3. Was ist -Kontraktion, und warum ist es zulässig?

4. Wann verwendet man foldr, wann foldl, und unter welchen Bedingungen ist das Ergebnis das gleiche?

1. fold r ist die kanonische einfach rekursive Funktion“ (Vorlesung). Was bedeutet das, und warum ist das so? Für welche Datentypen gilt das?

2. Wann kann fold r f a xs auch für ein zyklisches Argument xs (bspw. eine zyklische Liste) terminieren?

3. Warum sind endrekursive Funktionen im allgemeinen schneller als nichtendrekursive Funktionen? Unter welchen Voraussetzungen kann ich eine Funktion in endrekursive Form überführen?

1. Was ist ein abstrakter Datentyp (ADT)?

2. Was sind Unterschiede und Gemeinsamkeiten zwischen ADTs und Objekten, wie wir sie aus Sprachen wie Java kennen?

3. Wozu dienen Module in Haskell?

1. 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?

2. Warum finden Tests Fehler“, aber zeigen Beweise Korrektheit“, wie in der Vorlesung behauptet? Stimmt das immer?

3. Müssen Axiome immer ausführbar sein? Welche Axiome wären nicht ausführbar?

1. Der Datentyp Stream ist definiert als data Stream = Cons (Stream) Gibt es für diesen Datentyp ein Induktionsprinzip? Ist es sinnvoll?

2. Welche nichtausführbaren Prädikate haben wir in der Vorlesung kennengelernt?

3. Wie kann man in einem Induktionsbeweis die Induktionsvoraussetzung stärken, und wann ist das nötig?

1. Warum ist die Erzeugung von Zufallszahlen eine Aktion?

2. Warum ist auch das Schreiben in eine Datei eine Aktion?

3. Was ist (bedingt durch den Mangel an referentieller Transparenz) die entscheidende Eigenschaft, die Aktionen von reinen Funktionen unterscheidet?

PI-Fragen

Sebastian Benkel

January 2017

1 Introduction