

# Theorie der Programmierung I, WS 03/04

## Leitfaden zur Prüfungsvorbereitung

### Vorbemerkung:

Die folgenden Fragen sollen ein "Leitfaden" zur Prüfungsvorbereitung sein, d.h. diese Fragen sollte man sich selbst beim Lernen stellen und man sollte sie natürlich in der Prüfung beantworten können. Es handelt sich hier aber **nicht** um einen vollständigen Fragenkatalog, d.h. in der Prüfung können durchaus auch andere Fragen gestellt werden, oder sie können in anderer Form gestellt werden, oder es wird bei manchen Fragen nachgehakt, nach Beispielen bzw. Gegenbeispielen gefragt ...

## Kapitel I: Funktionale Programmiersprache

Typsystem:

- Wozu braucht man überhaupt ein Typsystem für eine Programmiersprache, warum arbeitet man nicht einfach mit ungetypten Sprachen?
- Was ist eine Typumgebung  $\Gamma$ ?
- Wie sehen Typurteile aus?
- Wie sehen die (wichtigsten) Typregeln aus: (APP), (COND), (LET-VAL), (ABSTR), (REC).
- Wie sieht der Algorithmus zur Typüberprüfung aus?
- Wie kommen die Bindungen in den Regeln (LET-VAL), (ABSTR), (REC) zum Ausdruck?
- Wie benutzt man die **rec**-Ausdrücke zur Deklaration rekursiver Funktionen?

Small step Semantik:

- Was ist ein Wert  $v$  (formale Definition, intuitive Vorstellung)?
- Wie sieht ein small step aus?
- Wie sehen die small step Regeln aus? Man sollte das Prinzip der Regeln verstehen. Dann kann man eine Regel auch selbst rekonstruieren, wenn man sie nicht mehr im Kopf hat.
- Wie spielen die verschiedenen Regeln zusammen, z.B. die drei Regeln für die Applikation.
- Wo kommt in den Regeln das call-by-value-Prinzip zum Ausdruck? Wie müsste man die Regeln abändern, um call-by-name zu erhalten?

- Kann man die Kernsprache noch weiter reduzieren, indem man ‘let’ oder ‘if-then-else’ als syntaktischen Zucker auffasst?
- Wie kommen exceptions ins Spiel? Wo braucht man sie überhaupt?
- Welche Ergebnisse haben wir über die small step Semantik bewiesen? (Determinismus, Preservation, Progress, Typsicherheit). Auch die Beweisideen sollte man kennen.

Big step Semantik:

- Wie sehen die big step Regeln aus? Auch hier sollte man das Prinzip der Regeln verstehen, um sie eventuell rekonstruieren zu können.
- Wie bringt man hier die exceptions ins Spiel?
- Wie hängen small- und big step Semantik zusammen?
- Wie kommt Divergenz bzw. Steckenbleiben in der big step Semantik zum Ausdruck?

Spracherweiterungen:

- Wie fügt man Listen zur Sprache hinzu? Wie muss man die Definitionen, Regeln, Sätze, Beweise ergänzen oder verändern?

## Kapitel II: Programmverifikation

- Definition der partiellen bzw. totalen Korrektheit.
- Wie sieht die formale Logik für totale Korrektheit aus? Wichtig sind natürlich nur die “returns”-Formeln, alles andere ist “übliche” Prädikatenlogik.
- Typregeln für die Logik, insbesondere für den returns-Ausdruck.
- Beispiel einer Spezifikation, etwa für die Fakultätsfunktion.
- Wichtigste Regeln des Kalküls: (COND-TRUE), (COND-FALSE), (APP), (BETA-V), (LET-VAL), (REC)
- Wie kommt man mit der (REC)-Regel überhaupt voran, obwohl ihre Voraussetzung größer ist als die Behauptung?