# Theorie der Programmierung I
## SS 2010
## Leitfaden für die Prüfungsvorbereitung

Die folgenden Fragen sollen ein „Leitfaden“ zur Vorbereitung auf die Prüfung sein, das heißt diese Fragen sollte man sich selbst beim Lernen stellen, und folglich in der Prüfung beantworten können. Das bedeutet nicht, dass in der Prüfung nicht auch andere Fragen gestellt werden können, oder diese Fragen in abgewandelter Form. Es handelt sich nicht um einen vollständigen Fragenkatalog, den es gilt, auswendig zu lernen.

**Syntax und small step Semantik**

- Wie sind die gültigen Ausdrücke definiert?
- Was ist ein Wert $v$ (formale Definition, intuitive Vorstellung)?
- Wie sieht ein small step aus (formale Definition, Beispiel)?
- Wie sehen small step Regeln aus? Nach welchem Kriterium kann man small step Regeln in zwei Kategorien einteilen? Man sollte hierzu das Prinzip der Regeln verstanden haben, so dass man nicht alle Regeln auswendig lernt, sondern viel mehr diese anhand des Prinzips konstruieren kann.
- Wie spielen die unterschiedlichen Regeln zusammen? Zum Beispiel die vier Regeln für die Applikation. Nennen Sie ein Beispiel für einen Ausdruck für dessen Berechnung die Regel (App-Left) benötigt wird.
- Wie äußert sich das *call-by-value* Prinzip? Wie müsste man das Regelwerk abändern um eine *call-by-name* Semantik zu erhalten?
- Nennen Sie Beispiele für abgeleitete small step Regeln für den syntaktischen Zucker. Begründen Sie, warum es sich um abgeleitete Regeln handelt.
- Kann man jedem (wohlgetypten) Ausdruck durch die small step Semantik einen Wert zuordnen? Begründen Sie Ihre Antwort.
- Nennen Sie die vier Möglichkeiten für Berechnungen. Geben Sie jeweils einen Beispielausdruck an.
- Welche wichtige Eigenschaft besitzt unsere Semantik? (Determinismus)

**Rekursion**

- Wie wird Rekursion in die Sprache eingebaut? (Syntax und Semantik)
- Geben Sie zwei Beispiele für rekursive Funktionen (z.B. `fact`, `fib`, ...).

**Typsystem**

- Welche Daseinsberechtigung haben Typsysteme? Warum arbeitet man nicht mit ungetypten Sprachen?
- Was ist eine Typumgebung $\Gamma$ (formale Definition, intuitive Vorstellung)? Was ist ein Typurteil?

- Wie sehen die wichtigsten Typregeln aus? Zum Beispiel (App), (Cond), (Abstr) und (Rec).
- Wie kommen die Bindungen in (Let), (Abstr) und (Rec) zum Ausdruck?
- Welche Eigenschaften der Sprache haben wir bewiesen? (Progress, Preservation, Safety). Erläutern Sie jeweils auch intuitiv die Bedeutung dieser Eigenschaften, und geben Sie die Beweisidee an.
- Geben Sie Beispiele für Restriktionen durch das Typsystem an. Also Ausdrücke, die zur Laufzeit nicht stecken bleiben, aber trotzdem nicht wohlgetypt sind.

**Typinferenz**

- Beschreiben Sie kurz, was Typinferenz ist, und welche Daseinsberechtigung es hat.
- Skizzieren Sie den Unifikationsalgorithmus und beschreiben Sie in eigenen Worten, was Unifikation ist.
- Was versteht man unter der allgemeinsten Lösung? (Intuition und formale Definition) Was versteht man unter dem allgemeinsten Typ?
- Beschreiben Sie die Arbeitsweise des Typinferenzalgorithmus anhand zweier Typregeln Ihrer Wahl. Gehen Sie dabei auch auf die Verbindung zum Unifikationsalgorithmus ein.
- Begründen Sie, warum der Unifikationsalgorithmus korrekt ist.

**Polymorphie**

- Was versteht man unter Polymorphie? Warum ist die Sprache $\mathcal{L}_3^{ti}$ nicht bereits polymorph?
- Wie sieht das Typsystem der polymorphen Sprache aus (polymorphe Typen, neue Typumgebungen, Typurteile und Typregeln)? Was versteht man unter dem *polymorphen Abschluss*.
- Wie sehen Beispiele für polymorphe Typen aus?
- Wie muss man den Typinferenzalgorithmus für Polymorphie erweitern?
- Wie zeigt man, dass die polymorphe Sprache nach wie vor typsicher ist?