# Prüfungsprotokoll Informatik A

| | |
|---|---|
| Vorlesungen: | Grundzüge der Informatik A1, A2 (bei Leonie Dreschler-Fischer), A3/F1 (bei Matthias Jantzen), A4 (bei Arno Rolf) |
| Prüferin: | Leonie S. Dreschler-Fischer (Beisitzer: Ralf Möller) |
| Prüfling: | Stefan Witt |
| Zuhörer: | 3 Kommilitonen |
| Datum: | 7.10.1999 ($9^{45}$ Uhr in R-111) |
| Vorbereitung: | 13 Wochen (intensiv) |
| Literatur: | A1-, A2-Skript von Leonie Dreschler-Fischer<br>A3-, F1-Skript (erste Auflage) von Matthias Jantzen<br>Ian Holyer: Functional Programming With Miranda<br>David Flanagran: Java In A Nutshell<br>Uwe Schöning: Theoretische Informatik - kurzgefaßt<br>Arno Rolf: Grundlagen der Organisations- und Wirtschaftsinformatik (auszugsweise)<br>Duden Informatik |
| Note: | 2,3 |

Leonie Dreschler-Fischer (LDF.): Mit welchem Semester möchten Sie beginnen?
Ich (I.): Mit dem ersten, A1 also.
LDF.: Was ist die Semantik in einer funktionalen Programmiersprache?
I.: Wertesemantik; Ausdrücke werden auf Werte reduziert, und der Wert ist die Bedeutung des Audrucks; es gibt keine Seiteneffekte.
LDF.: Warum will man Seiteneffekte vermeiden?
I.: Seiteneffekte erhöhen die Komplexität der Zusammenhänge von Objekten, so dass Korrektheitsbeweise schwieriger werden.
LDF.: Hat ein Ausdruck wirklich immer den gleichen Wert, gibt es da keine Ausnahme?
I.: Doch, in Miranda gibt es Ströme, mit denen z.B. Tastatureingaben realisiert werden; ansonsten ist die referenzielle Transparenz in funktionalen Programmiersprachen wie Miranda oder LISP gewährleistet.
LDF.: LISP ist nicht referenziell transparent.
I.: Ja, LISP ist keine reine funktionale Programmiersprache; die referenzielle Transparenz ist nicht gewährleistet; man kann LISP aber funktional programmieren, so dass sie gewährleistet ist.
LDF.: Welche Reduktionsstrategien gibt es?
I.: vorgezogene und verzögerte Auswertung (erklärt)
LDF.: Vor- und Nachteile? Geben Sie ein Beispiel an, und erläutern Sie die Auswertung daran.
I.: `square (4 + 4)` aufgeschrieben und innere/äußere Reduktion, vorgezogene/verzögerte Auswertung erklärt
LDF.: Ist in LISP die Auswertestrategie immer die vorgezogene Auswertung?
I.: Nein, bei Special-Form-Ausdrücken nicht, z.B. if-Konstrukte (erklärt).
LDF.: Was für Probleme gibt es bei der Auswertung von Variablen?
I.: Probleme? Eine Variable hat einen Wert, und der wird einfach ausgelesen.
LDF.: Kann eine Variable auch mehrere Werte haben?
I.: Ja, in LISP gibt es den Symbol-Value und den Function-Value (Referenz mit #'); beide können an ein Symbol (eine Variable) gebunden werden.
LDF.: So speziell wollte ich das nicht wissen; wie sieht es aus, wenn ich mehrere Variable mit gleichem Namen habe?
I. *(darauf will sie hinaus!)*: Globale Variable (Special Variables) können durch lokale Variable verschattet werden (erläutert).
LDF.: Geben Sie ein Beispiel für eine gecurryte Funktion an.
I.: `(+ 1)` aufgeschrieben, erklärt
LDF.: Ich meinte in LISP.
I.: `(defun pluseins (x) #'(lambda (x) (+ 1 x)))` aufgeschrieben, ohne zu merken, dass ich Inkrement- und Addierfunktion gemischt habe
LDF.: Schauen Sie doch nochmal genau hin.
Ralf Möller: Da muss n stehen! *(Bei der Notenvergabe erklärt Ralf Möller mir nochmal ganz ausführlich, wie die Bindung funktioniert, obwohl mir das schon klar war - offenbar hat mir dieser Fehler die Note in den Sand gesetzt)*
I.: Ach so, `(defun plusn (n) #'(lambda (x) (+ n x)))`
LDF.: Ja, und wie sieht das mit den Variablenbindungen aus?
I.: n ist nur von der anonymen Lambda-Funktion referenzierbar, sie ist in der Closure gekapselt
LDF.: Ja. Wechseln wir zu A2. Was kennzeichnet die objektorientierte Programierung?
I.: Modellierung von Objekten, Zusammenfassen zu Klassen, Vererbung von Methoden und Eigenschaften (erklärt)
LDF.: Ja, folgende 3 Eigenschaften sind wichtig: Klassifikation, Vererbung und Polymorphie (hatte ich nicht genannt). Was ist Polymorphie?

I.: Polymorphie von Methoden: Abstrakte generische Operation (auf mehrere Klassen polymorph anwendbar) wird

definiert, vererbt und durch klassenspezifische Methoden implementiert.
LDF.: Nehmen Sie an, Sie haben 2 Softwarepakete, die sie spezialisieren wollen. Was tun Sie? Geben Sie ein Beispiel an.
I.: ???
LDF.: Nehmen wir ein maritimes Beispiel (erklärt Klassen Schiff, Segelschiff und Motorschiff); wenn sie aber einen Motorsegler haben, der Segelschiff und Motorschiff ist, was tun Sie?
I.: Ich definiere eine eigene Klasse Motorsegler, die von Segel- und Motorschiffen erbt (Mehrfachvererbung).
LDF.: Wenn Sie aber die Methoden der beiden Oberklassen teilweise mitbenutzen, teilweise weiter ergänzen wollen, wie tun Sie das?
I.: Durch Ergänzungsmethoden (in CLOS mit :before und :after) und den Super-Call (Methode der Oberklasse aufrufen); erklärt.
LDF.: Werden wir ganz konkret. Wenn in der Klasse Schiff generische Operationen vorwärts-, rückwärtsfahren und anlegen definiert sind und in den Unterklassen Segel- und Motorschiff implementiert sind und wenn Anlegen in der Motorseglerklasse implementiert ist, welche Methode wird beim Anlegen des Motorseglers aufgerufen?
I.: Die speziellste, bei den Motorseglern also die klassenspezifische Methode.
LDF.: Ja, und die benutzt die Methoden der übergeordneten Klassen mit, obwohl die Programmierer der Oberklassen-Methoden sie nicht dafür gedacht haben.
LDF.: Kommen wir zu A3. Sie haben sich sicher mit Logik beschäftigt. Was ist die Aussagenlogik?
I.: Der Kalkül der Aussagenlogik besteht aus der syntaktischen Basis der aussagenlogischen Formeln (mit Junktoren, Variablen und Konstanten) und der zugehörigen Semantik.
LDF.: Ist die Aussagenlogik entscheidbar?
I.: Ja, man muss allerdings die Variablen durch 1 und 0 (bzw. True und False) ersetzen, um aus einer Aussageform eine Aussage zu erhalten.
LDF.: Und wie sieht es mit der Prädikatenlogik aus?
I.: Die Prädikatenlogik ist nicht entscheidbar. Man kann aber eine erfüllbarkeitsäquivalente Prädikatenformel finden.
LDF.: Wieso benutzt man die Aussagenlogik, obwohl sie unentscheidbar ist?
I.: Die Aussagenlogik ist doch entscheidbar!
LDF.: Äh, die Prädikatenlogik meinte ich.
I.: In der Aussagenlogik kann man keine Quantoren »es gibt« und »für alle« benutzen, die man aber oft benötigt.
LDF.: ...und die Prädikate. In A3 haben Sie sich sicher ausführlich mit formalen Sprachen beschäftigt. Bitte geben Sie die Chomsky-Hierarchie an.
I.: skizziert und erklärt: reguläre Sprachen (endlicher Automat), (deterministische) kontextfreie Sprachen (Kellerautomat), kontextsensitive Sprachen (linear beschränkter Automat), entscheidbare Sprachen, Phrasenstruktursprachen (Turing-Maschine), abzählbare Sprachen
LDF.: Und wo liegen die aufzählbaren Sprachen?
I.: Die sind äquivalent zu den Typ-0-Sprachen.
LDF.: Können Sie das beweisen?
I.: (ergebnislos überlegt); die Menge aller Wörter ist auf jeden Fall aufzählbar...
LDF.: Geben Sie eine Turing-Maschine an, die die aufzählbaren Sprachen erzeugt. Oder denken Sie an ein PASCAL-Programm, das das leistet *(nach der Prüfung habe ich festgestellt: Die Menge aller aufzählbaren Sprachen ist überabzählbar, eine Turing-Maschine kann immer nur* eine *aufzählbare Sprache zu einer Typ-0-Grammatik erzeugen (vielleicht habe ich mir die Frage falsch gemerkt, oder sie war falsch gestellt); im Jantzen-Skript wird von einer NTM ausgegangen, was viel einfacher ist als mit einer DTM bzw. einem deterministischen PASCAL-Programm).*
I.: (überlegt) Der Ableitungsbaum von Typ-0-Grammatiken kann aber unendlich sein...
LDF.: Ja, aber ist das eine Schwierigkeit?
I.: Nein, z.B. $\{a,b\}^*$ ist lexikalisch aufzählbar (erläutert).
LDF.: Sie müssen einfach alle Produktionen durchgehen. Sie haben gesagt, die kontextsensitiven Sprachen sind Teilmenge der entscheidbaren Sprachen. Woran liegt das?
I.: An der Monotonie; der Ableitungsbaum ist endlich, und so kann effektiv jedes Wort auf seine Zugehörigkeit zu den kontextsensitiven Sprachen überprüft werden.
LDF.: Was für ein Typ ist $a^n\ b^m\ c^n$?
I.: kontextfrei, Grammatik dazu: `S -> aSc | A, A -> Ab | lambda`
LDF.: Welcher Automat akzeptiert die Sprache?
I.: Ein Kellerautomat, in diesem Fall ein deterministischer.
LDF.: Erläutern Sie, wie der Kellerautomat arbeitet.
I.: as lesen und kellern, bs lesen und cs lesen und mitzählen, ob es genausoviele sind, wie as auf den Keller geschrieben wurden
LDF.: Wie erkennt der Automat, dass ein Wort nicht zur Sprache gehört?
I.: Wenn das erste b gelesen wurde, darf kein a mehr gelesen werden, entsprechend für bs; Test, ob der Keller zum Schluss leer ist usw.
LDF.: Dann machen wir 'mal an dieser Stelle Schluss.