

# Grundlagen Informatik Vordiplom

Manuel Klimek

gehalten am 17.03.2000

<b>Prüfer</b>	Prof. Dr. Lengauer
<b>Note</b>	1.3

## 1 Ablauf

Prof: Welche Scheine haben sie gemacht?

Ich: Info 1, Info 2 hab ich nicht mitgeschrieben.

Prof: Was hat ihnen denn am besten gefallen?

Ich: Formale Sprachen!

Prof: (Ist ganz entsetzt, an diesem Tag hatten vor mir schon fast alle anderen auch formale Sprachen gewählt. Er fragte, ob wir uns abgesprochen hätten...)

Prof: (Sichtlich genervt) Ok, zeichnen sie mal die Chompsky-Hierarchie auf!

Ich: (Male sie hin, erkläre alles mögliche dazu...)

Prof: So, welches sind denn hier die wichtigsten Sprachen?

Ich: Kontextfreie, mit kontextsensitiven Nebenbedingungen, da diese in Programmiersprachen verwendet werden.

Prof: Warum nimmt man da nicht gleich kontextsensitive?

Ich: Das Parsen ist zu aufwendig. Bei Programmiersprachen ist es sinnvoll, in einem ersten Schritt den Ableitungsbaum mit einer kontextfreien Sprache zu parsen, um dann erst auf Nebenbedingungen zu prüfen.

Prof: Was für Probleme gibt es denn beim parsen?

- Ich: (hermumtu..., dann mit ein bisschen Hilfe) Mehrdeutigkeit
- Prof: Was genau ist denn Mehrdeutigkeit?
- Ich: (Will es erst anschaulich erklären, doch Lengauer besteht auf einer genauen Def.) Wenn an einer Stelle beim Parsen mehrere Ableitungsregeln anwendbar sind (das hat ihm dann anscheinend gepasst...)
- Prof: Schreiben sie mal eine eindeutige Grammatik hin, die Summe und Produkt parsed.
- Ich: (Habe erst mal Produkt und Summer vertauscht ... peinlich, peinlich)  
 $S ::= P + S$   
 $P ::= Z + P$  ( Z ist Zahl )
- Prof: Was ist hier besonderes an ihrer Grammatik?
- Ich: (Ich check mal wieder gar nix, er hilft wieder drauf, läßt mich den Ableitungsbaum hinmalen). Hier wird von rechts nach links berechnet.
- Prof: (Hier fand Lengauer irgendwie einen sehr geschickten Übergang zur imperativen Programmierung, der mir leider nicht mehr einfällt) - Was hat ihnen besser gefallen, imperative oder funktionale Programmierung?
- Ich: Imperative konnte ich schon, ist mir also leichter gefallen... :)
- Prof: (Lengauer war sichtlich belustigt über diese Antwort). Nein, sie müssen mir jetzt schon sinnvolle Argumente bringen!
- Ich: Je nach Problem ist mal die funktionale, mal die imperative Programmierung sinnvoller. Funktionale bei mathematischen, abstrakten Problemen, imperative Programmierung bei effizienz-Anforderungen.
- Prof: Was ist denn ein typisches Problem, das besser imperativ gelöst wird?
- Ich: Betriebssystem.
- Prof: (Ging jetzt noch genauer auf Vorteile von imperativer bzw. funktionaler Programmierung ein, wollte auch von mir noch mehr wissen. Ich habe jedoch nur geschickt drumherumgeredet, also ist er zum nächsten Thema)...
- Prof: (Hier weiss ich wirklich nicht mehr genau, was er gesagt hat... Ich wusste nur worauf er hinaus wollte, da ein Mitstudent mir eine halbe Stunde vor der Prüfung den Tip gegeben hat, mir das noch zu merken. Es ging

um die Interpretationsfunktion bei der Imperativen Programmierung, genauer gesagt, deren Funktionalität.

Ich: Sei A das Alphabet, B die Menge der Zustände  
I:  $A \rightarrow (B \rightarrow B)$   
(noch ein bisschen drumherumerklär...)

Prof: Und wie sieht das im Funktionalen aus?

Ich: (Hier hab ich lange überlegt, das hatte ich nämlich gar nicht gelernt, hatte dann auch noch einen kleinen Fehler drin, bevor ich zu folgender Funktionalität gelangt bin)  
I:  $A \rightarrow (B \rightarrow B)$   
mit  $B = \{true, false\}$ .

Prof: (Jetzt hat er auch wieder geschickt das Thema gewechselt...) - Was für Parameterübergabemechanismen gibt es denn?

Ich: (Male die Tabelle aus dem Script hin, erklär ein bisschen drumherum)

Prof: Schreiben sie mal eine Funktion auf, die sich für call-by-name und call-by-reference unterschiedlich verhält.

Ich: Also...

```
fun ( a, i ) {  
    i++;  
    a := 3;  
}
```

Aufruf:  
i = 1;  
fun ( a[i], i );

Ergebnis:  
Call-by-name: a[2] = 3;  
Call-by-ref: a[1] = 3;

Prof: (Er meinte jetzt, es ginge auch mit einem Parameter... Und einer globalen Variable). Was gibt es denn im Funktionalen für Übergabemechanismen?

Ich: Call-by-value und call-by-name.

Prof: Gibt es da einen Unterschied?

Ich: (Überlege ein bisschen) Nein, eigentlich nicht, da es ja keine Seiteneffekte gibt.

Prof: Dann sehen sie sich mal das hier an:  
if ( true, 1, 1/0 )

Ich: (Vordenkopfschlag... das ist natürlich bei call-by-value nicht definiert, bei call-by-name liefert es 1 zurück)

Prof: (Lacht erstmal... Dann kam er noch irgendwie auf die Berechenbarkeit, weiss aber nicht mehr, wie er das geschafft hat. Nur an eine Frage kann ich mich noch erinnern) Gibt es ein endliches Turing- Programm, das in endlicher Zeit entscheiden kann, ob eine terminierende Funktion terminiert?

Ich: Nein.

Prof: (Grinsbisüberbeideohren)

Ich: Shit!

Prof: (Will die Frage umformulieren...)

Ich: Erklär ihm einfach was ich weiss dazu...

Prof: Ok, die Zeit ist um, das war eine gute Leistung... Viel Spaß im Hauptstudium

Ich: :-) Yea!

## 2 Zusammenfassung

An diesem Tag ist einer von ungefähr 10 durchgefallen und die Noten gingen von 1.0 bis 4.3. Meiner Ansicht nach ist er kein sehr schwerer Prüfer, aber man darf sich nicht darauf verlassen, dass er nur Fragen zu dem gewünschten Gebiet stellt. Bei mir ist (vom zeitlichen Aufwand her) Formale Sprachen ziemlich kurz gekommen... Und imperative Programmierung habe ich nur noch aus dem Grundwissen gezogen. Dafür, wie oft ich nicht genau bescheid wusste, ist die Note sehr großzügig geworden...