Prüfer: Prof. Dr. Rüdiger Valk

Beisitzer: Dr. Bernd Kirsig

Datum/Zeit: 26.1.1999 / 10:00 Uhr (Y2K-compliant!)

Extension: 12 Grad Celsius, regnerisch

Dauer: 35 Minuten

Zuhörer: 4

Note: 2,7 (angesichts der Vorbereitungszeit etwas enttäuschend)

Vorbereitung: 2 Jahre, davon 3 Wochen intensiv (12 Stunden am Tag)

Allgemeines

Dieses Protokoll ist vermutlich nur für Studis interessant, die ihr Studium vor 1998 angefangen haben, da sich durch die Reformierung des Studiums sicherlich auch einiges bei den Prüfungen ändern wird. Es bleibt ernsthaft zu hoffen, daß die Stoffmenge sich dadurch reduziert. Die Anforderungen entsprechen meiner Meinung nach nicht der Wichtigkeit des Themenkomplexes.

Material

- Jantzen: Terroristische Grundlagen der Programmierung
  (inakzeptabel fehlerhaft)
- Valk/Jessen: Modelle für Rechensysteme (mindestens ohne Kapitel 3 und 7)
  (sehr formal, manchmal fehlen Zusammenhänge, aber obligatorisch)
- Schöning: Terroristische Informatik — kurzgefaßt
  (eine Oase für gefrustete Studis, ich habe Professor Schöning eine kleine Dankmail an die Uni Ulm geschickt)
- Knuth/Graham/Patashnik: Concrete Mathematics
  (noch unterhaltsamer als die Von-Der-Heide-Show im Grundstudium)
- Der LaTeX-Begleiter
  (für die griechischen Symbole)
- Prüfungsprotokolle
  (wetten, daß Du auch keins schreiben wirst?)
- Quake II
  (ich habe mich sowieso zuwenig entspannt)
- große Mengen Koffein
  (bbeerruuhhiiggtt eeiinneenn iirrggeennddwwiiee nniicchhtt)

Vorbereitung

Es ist wohl unbestritten, daß die Theorieprüfung zu den härtesten Prüfungen zählt. Alleine alle Inhalte — sofern befriedigend verstanden — ins Hirn zu bekommen, geht einem schon arg an die Sättigungsfüllung. Um die Prüfung trotzdem unbeschadet zu überstehen, finde ich folgende Punkte sehr wichtig:

- Keine Panik!
  Der Stoff ist extrem umfangreich und es werden sogar alle Grundlagen aus A3 implizit verlangt. Allerdings gibt es kaum etwas, was man nicht in aller Ruhe verstehen könnte. Bei so mancher Herleitung (Little oder Füllung bei M/M/m/k) braucht man nur die richtige Idee und schon wird vieles klar und leicht zu lernen. Ich habe wenig in den Vorlesungen verstanden und es trotzdem gemeistert. Achte vor allem auch auf allgemeine Bemerkungen, die zwischen den Definitionen stehen. Um endlich die nötige Ruhe für die Vorbereitung zu haben, habe ich mir vier Wochen Urlaub vom Job geleistet.
- Prüfungsprotokolle lesen
  Es ist gar nicht möglich, alle Inhalte genau zu kennen. Einige Gedanken sollte man sich aber vor der Prüfung schon gemacht haben. Wenn Du die Literatur auswendig kannst, ist noch keine gute Note gesichert. Rechensysteme und TGP hängen stark zusammen (Serialisierbarkeit ist entscheidbar und NP-vollständig) und die Zusammenhänge sind nicht immer offensichtlich. Da können die Prüfungsprotokolle schon einiges bieten.
- Mitlerner
  Ich bin normalerweise ein Einzelgänger, weil es viel Overhead kostet, sich mit anderen Kommilitonen auszutauschen. Das ist aber bei Theorie unumgänglich. Nirgendwo anders ist der Unterschied zwischen Verstehen und Wiedergeben so groß wie hier. Außerdem muß man sich motivieren, mit den anderen mitzuhalten. Such Dir aber jemanden, der in der gleichen Zeit die Prüfung machen will. Mein Mitlerner hatte leider zu großen privaten Overhead und so war ich am Ende fast alleine. In der Endphase vor der Prüfung sollte man auch einige Prüfungssimulationen abhalten.
- Karteikarten o.ä.
  Wenn Du kurz vor der Prüfung stehst, sind eigene Aufzeichnungen von Vorteil. Man kann nicht ‚mal eben' noch mal die Literatur durcharbeiten — das dauert mindestens zwei Tage (wenn man alles schon verstanden hat). Manche Begriffe (z.B. Funktionseinheit, Interpretation) kann man nur schwer in Worte fassen. Überleg Dir zu jedem Begriff einen Satz. Übersichten sind auch wichtig. Wer weiß schon, was abzählbar, aufzählbar oder entscheidbar ist und was welche Menge impliziert. Übersichten helfen einem auch bei „malen Sie doch mal auf'-Fragen. Ich habe übrigens für unsere Lerngruppe eine Sammlung von (hoffentlich eines Tages legendären) Lernkarten in LaTeX erstellt. Sie sind der Nachwelt erhalten geblieben und als Postscript-Dokument unter `ftp://www.newton-lifestyle.de/pub/theorie` saugbar.
- Mut zur Lücke
  Lerne alles, aber nicht alles auswendig. Ich habe so indiskret Lücken gelassen, daß ich einige Themen gar nicht gelernt habe, die ich für unwichtig gehalten habe. Leider war Valk da andere Meinung und so bin ich böse auf den Bauch gefallen. Manche Themen wie zellulare Automaten, neuronale Netze, Kosten/Nutzen in Rechensystemen scheinen aber wirklich nicht oberwichtig zu sein.

Übrigens liegt die Durchfallquote momentan bei 17,1% und die Durchschnittsnote bei 2,8. Das ist besser als bei der B-Prüfung! Die Wahrscheinlichkeit sei mir Dir. :-)

Prüfungsverlauf

Auch in meinem Protokoll möchte ich mehr den Charakter der Prüfung wiedergeben als nur die Fragen. Bei manchen Protokollen gewinnt man den

Eindruck, das Opfer hätte alles sofort perfekt gewußt. Es gibt meistens eine ganze Reihe von Nachfragen. Ich hoffe, das wird im folgenden deutlich.

Auftragssysteme

Valk Was hatten wir als Literatur verabredet?

$>$ Das TGP-Skript von Professor Jantzen und das Rechensysteme-Buch ohne die Kapitel 3 und 7.

Valk In Rechensystemen geht es ja vorne mit Auftragssystemen los. Was ist denn ein Auftragssystem?

$>$ Ein Auftragssystem ist ein Tupel $AS = (A, \prec)$, wobei die Präzedenzrelation irreflexiv und transitiv ist.

Valk Dann gibt es ja Kausalnetze. Wie sehen die aus?

$>$ Kausalnetze sind formal Tupel $N = (S, T, F)$. Und für die Flußrelation gilt hier, daß die transitive Hülle (*die Kausalrelation*) auch irreflexiv und transitiv ist, weil sie der Präzedenzrelation beim Auftragssystem entspricht.

Valk Wie hängen die denn damit zusammen?

$>$ Im Prinzip sind Kausalnetze und Auftragssysteme äquivalent.

Valk Und wie formen Sie die um?

$>$ Um vom Kausalnetz auf ein Auftragssystem zu kommen, nimmt man die Transitionen als Aufträge.

Valk Was gilt denn bei Kausalnetzen noch. Etwas fehlt da ja noch.

$>$ Ach ja, jede Stelle darf maximal eine Eingangs- und Ausgangstransition haben. Es darf also keine Konfliktstellen geben.

Valk Warum ist das so?

$>$ Die Kausalrelation gibt ja die kausale Abfolge der Transitionen im Netz an. Es gibt also bei einem Kausalnetz keine Entscheidungen der Wahl der Wege mehr.

Valk Wofür braucht man denn Kausalnetze zum Beispiel?

$>$ Zur Konstruktion von Prozessen für S/T-Netze.

Valk Und wie ist ein Prozeß definiert.

$>$ Er besteht aus einem markierten Kausalnetz N_K und aus einer Relation ... äh ... Abbildung p_K, die alle Stellen und Transitionen des Kausalnetzes auf die ursprünglichen Stellen des S/T-Netzes abbildet. Also:
$p_K : (S_K \cup T_K) \rightarrow (S \cup T)$

Valk Was müssen Sie denn noch fordern, wenn sie einen Prozeß aus einem S/T-Netz bilden?

$>$ Das Netz darf keine Kapazitäten mehr enthalten. Es muß also durch Einführung von Komplementärstellen kontaktfrei gemacht werden.

Valk Ja, und warum sind Prozesse nun Kausalnetze?

$>$ Weil ja die Ausführung der Transitionen festgelegt ist. Bei der Modellierung eines Prozesses gibt es ja auch keine Entscheidungen mehr.

Valk Schon, aber warum nicht.

$>$ Einen Prozeß liest man ja sozusagen von links nach rechts. Ein Prozeß gibt also die Abfolge der Transitionen wieder. (*...irgendwas mußte ich vergessen haben, aber was?*)

Valk Was stellt der Prozeß denn dar?

$>$ Die Verfolgung der Marken in einem S/T-Netz über die Zeit.

Valk Und warum darf es dann keine Konfliktstellen geben?

$>$ Weil es zum Zeitpunkt des Prozesses keine Entscheidungen mehr gibt. (*hatte ich zwar schon gesagt, aber vielleicht war es untergegangen*)

Valk Wie sieht das denn mit einer Schaltfolge aus?

$>$ Eine Schaltfolge ist eine Folge von Transitionen, die nacheinander schalten.

Valk Und im Prozeß müssen die auch nacheinander schalten?

$>$ Nein, die Reihenfolge kann beliebig sein.

Valk Was ist denn restriktiver, die Schaltfolge oder der Prozeß?

$>$ (*Bitte? Das ist doch in diesem Falle dasselbe. Er wollte aber darauf hinaus, daß das ja nur für einen bestimmten Prozeß zu einem S/T-Netz gilt. Schließlich gibt es ja eventuell mehrere Prozesse, wenn das S/T-Netz noch Konfliktstellen hat.*)

Valk Na gut, was ist denn ein schematisches Auftragssystem?

$>$ Es besteht auch wieder aus einer Auftragsmenge und einer Präzedenzrelation. Gefordert werden hier auf jeden Fall die direkten Präzedenzen $l_i \prec s_i$. Und es kann noch weitere Präzedenzen $s_i \prec l_j$ geben. Außerdem gibt es eine Menge von globalen Variablen und jeder Auftrag hat noch eine Menge von Eingangs- und Ausgangsvariablen.

Valk Warum ist denn ‚schematisch'?

$>$ Man untersucht daran nur die Zusammenhänge und Einflüsse von Aufträgen. Die eigentliche ‚Natur' der Aufträge ist hier nicht wichtig.

Valk Aber wenn wir die Variablen kennen, haben wir doch auch die Einflüsse.

$>$ Man kann die konkrete Festlegung der Aufträge nur bei einer Interpretation betrachten.

Valk Darauf wollte ich hinaus. Was ist denn eine Serialisierung?

$>$ Eine Ausführungsfolge, die seriell ist. l_i und s_i kommen in der Ausführungsfolge direkt hintereinander.

Valk Was heißt denn, daß zwei Ausführungsfolgen äquivalent sind?

$>$ Daß für alle Interpretationen das gleiche Ergebnis herauskommt.

Valk Ist das entscheidbar?

$>$ Ja, das ist entscheidbar.

Valk Und wie?

$>$ Es gibt ja noch die Definition, daß zwei Ausführungsfolgen äquivalent sind, wenn die Menge der relevanten Aufträge identisch ist und die Ausführungsfolgen (*Flüchtigkeitsfehler!*) gleich sind.

Valk Aber wir betrachten doch nur eine einzige Ausführungsfolge?!

$>$ Ähm, natürlich die Werteübertragungsrelation und nicht die Ausführungsfolge.

Bedienstrategie FCFS

Valk Genau. Gut, dann geht es ja im Buch noch um Bedienstrategien. Wissen Sie denn auch den Erwartungwert der Wartezeit für die Strategie FCFS?

$>$ Ja, die Pollaczek-Khinchinsche Mittelwertformel. $E[W] = \frac{\rho \cdot E[B^2]}{2 \cdot (1-\rho)}$ (*Es gab keine großen Nachfragen. Merkwürdig* ...)

Valk Richtig. Und was ist dabei ρ?

$>$ Die Auslastung.

Valk Und wie ist die definiert?

$>$ $\rho = \frac{\lambda}{m \cdot \mu}$ (*das war hier die falsche Formel*)

Valk In welchem Wartesystem betrachten wir denn die Bedienstrategien?

$>$ In M/G/1.

Valk Wir haben also nur eine Bedieneinheit. Wie kann es denn dann ein μ geben? (*ich würde mich eher fragen, warum es dann ein m geben kann, aber egal*)

$>$ Ach nein. $\rho = \frac{d}{c}$

Sättigungsfüllung

Valk Ja. Wie ist denn die Sättigungsfüllung definiert?

$>$ (*Auweia, eine der Sachen, die ich zwei Tage vor der Prüfung erst verstanden hatte — dachte ich wenigstens.*) $\overline{f}_S = \overline{b} \cdot c$

Valk F quer? Ist denn hier die mittlere Füllung gefragt?

$>$ (*Wacker voran, mutiger Krieger* ...) Ach nein, der Strich über dem f kann weg.

Valk Und bei der Bedienzeit?

$>$ Äh ...

Valk Erinnern Sie sich noch an die Zeichnung im Buch?

$>$ Ja. (*Zeichnung von Seite 387 in Rechensystemen korrekt gezeichnet, aber leider die Diagonale als Verweilzeit beschrieben*)

Valk Wieso ist das denn die Verweilzeit?

> Äh . . .

Valk Sie haben hier ja zwei Asymptoten. Wie sieht denn nun die Verweilzeit aus?

> (*ich war so gestreßt, daß mir die Graphen von Seite 388 nicht wieder einfielen*). Keine Ahnung.

Valk Die Verweilzeit verläuft natürlich darunter. Ich sehe schon, in dem Gebiet sind sie nicht so fit. (*Na, toll . . .*) Können Sie mir denn wenigstens etwas zu den Verweilzeitgesetzen sagen? Oder sollen wir das lieber auch überspringen?

> (*Bevor ich noch mehr versiebe, wollte ich lieber den Bereich loswerden und noch ein paar meine genial gelernten Herleitungen vorführen.*) Am besten ja.

RAM

Valk Also gut, lassen wir das mal. Kommen wir mal zu TGP. Was ist denn eine RAM?

> Eine RAM ist eine Random Access Machine, ein Modell für einen sequentiellen Computer. Dazu gehört eine Anzahl von Registern, wobei das unterste Register auch Akkumulator heißt. Der Index der Register wird Adresse genannt. Und es gibt ein festgelegtes Programm und je ein Eingabe- und Ausgabeband.

Valk Wie können Sie denn eine RAM veranschaulichen?

> (*Huch? Wo stand das denn? Egal, kleine Zeichnung mit zwei Bändern und einem Stapel von Adressen.*)

Valk Und ist die genauso mächtig wie die Turing-Maschine?

> Ja.

Valk Wie können die sich simulieren?

> (*Stimmt, ich wollte mir ja noch mal die Simulation vor der Prüfung angucken. Hmmm, etwas spät vielleicht.*)

Valk Okay, nehmen wir mal eine Turing-Maschine dazu (*er zeichnete eine mit einem Band und einer endlichen Kontrolle*). So jetzt haben Sie beides. Wie können Sie denn da jetzt die Simulation durchführen?

> (*Jede noch so gut gemeinte Hilfe war leider umsonst. Jetzt wurde ich doch sehr nervös. Schon das dritte Thema, wo mir nichts mehr einfiel.*)

Valk Na gut, was ist denn eine PRAM?

> Ein Modell für einen parallen Computer. Wir haben da mehrere Prozessoren, die jeweils wieder RAMs sind.

Valk Sind denn PRAMs mächtiger als RAMs?

> (*Guuuute Frage! Ich bezog es leider nicht auf die Sprachen und meine Anschauung sagte mir . . .*) Ja.

Valk RAMs akzeptieren ja die aufzählbaren Sprachen. Und die PRAMs?

$>$ Auch die aufzählbaren Sprachen. Die sind in bezug auf die Sprachen natürlich gleichmächtig.

Valk Wie sieht denn das aus, wenn wir mehrere Prozessoren haben?

$>$ (*Die Frage war mir irgendwie zu allgemein. Also alles rauskramen, was einem einfällt.*) Es kann zu Konflikten bei Schreibvorgängen kommen. Deshalb gibt es CRCW, CREW und EREW. (*mit den Unterklassen CRCW-pri, CRCW-com und CRCW-arb erklärt*)

Valk Nun ist die Anzahl der Prozessoren ja dynamisch. (*ist sie das?*) Wie sieht denn das aus, wenn wir eine polynomielle Anzahl von Prozessoren haben?

$>$ (*WAS? Herr Baron, die Kugel bitte.*) Keine Ahnung.

P / NP

Valk Na gut, lassen wir das auch mal. (*So langsam beschlich mich das Gefühl, daß meine 4,0 ins Wanken kam.*) Was ist denn P?

$>$ Die Menge aller Sprachen, die von einer deterministischen polynomiell zeitbeschränkten Turing-Maschine akzeptiert wird.

Valk Und NP?

$>$ Dasselbe für nichtdeterministische Turing-Maschinen.

Valk Und was heißt NP-vollständig?

$>$ Wenn ein Problem NP-vollständig ist, dann heißt es, daß das Problem ein Element von NP ist und sich jedes Problem aus NP auf dieses Problem reduzieren läßt. (*habe ich formal gemacht*)

Valk Was heißt reduzierbar?

$>$ Man kann ein Problem auf ein anderes abbilden. Es gibt also eine Funktion f, wobei: $\forall x \in L_1, y \in L_2 : \ f(x) = y$

Valk Muß vor dem y ein Existenz- oder Allquantor stehen?

$>$ Für jedes x *gibt es* ein y.

Valk Das ist alles?

$>$ Nein, wichtig ist noch, daß die Funktion f von einer Turing-Maschine ausgeführt werden kann. Es muß also also $z_0 x \vdash^* z_e y$ gelten.

Valk Das stimmt nicht ganz, aber das reicht hier. (*Was fehlte denn noch?*) Wenn wir jetzt zwei Probleme haben, was heißt denn, daß L_1 auf L_2 reduzierbar ist? (*Mir fiel nichts weiter dazu ein.*) Na, das bedeutet doch, daß wenn wir L_2 lösen können, dann auch L_1.

$>$ Äh, ja, natürlich. (*hatte ich sowas nicht gesagt?*)

Valk Okay, hören wir auf.

Nachbesprechung

Prof. Valk äußerte sich etwas enttäuscht darüber, daß das mit dem RAMs nicht so gut geklappt hat. Immerhin sei das doch in der Vorlesung ausführlich erklärt worden. (Die Vorlesung fand ich so kryptisch, daß ich sie dreimal gehört habe und mir auch das nichts gebracht hat.) Und meine Unkenntnis über die Simulation von RAMs und Turing-Maschinen sowieso der Komplexität von PRAMs ging wohl über seine Schmerzgrenze. Somit erhielt ich eine 2,7. Mein primäres Ziel war, die Prüfung überhaupt zu bestehen. Nachträglich bin ich aber über das ‚ungute' Ergebnis nach der langen Vorbereitungszeit doch etwas enttäuscht. Leider habe ich aber meine Lücken doch etwas unausgeglichen gewählt und auch freiwillig zuwenig erzählt.

Endzustand

Dieses Protokoll sei allen gewidmet, die mit erstaunlicher Gelassenheit meine teilweise dummen Fragen ertragen haben und eine Woche vor der Prüfung so manchen Angstanfall mit viel Koffein und gutem Zureden beenden konnten. Ihr da draußen, die es noch vor sich haben: bei vernünftiger Vorbereitung ist es keine Frage des ‚Bestehens' sondern nur der Note. Gute Nerven und viel Erfolg!