```
Gedächtnisprotokoll Diplomprüfung Theoretische Informatik

Datum:    05.08.2006, 10:30
Prüfer:   Vöcking - Effiziente Algorithmen, Vöcking WS 04/05
          Noll    - Programmanlyse und Compileroptimierung, Indermark WS 04/05
          Noll    - Funktionale Programmierung, Giesl SS 05
Prüfling: Christopher Vogt (mailto:theoprotokoll@cvogt.org)
Note:     1,3

Atmosphäre:
Wie alle immer sagen: sehr angenehm und freundlich. Die Nervösität gibt sich
mit jeder richtigen Antwort mehr und mehr.

Bemerkungen vorweg:
Ich habe das Protokoll geschrieben soweit ich mich erinnere. Die Reihenfolge u.
genaue Fragestellung kann also natürlich abweichen und es kann auch sein, dass
ich etwas vergessen habe. Teilweise habe ich das Protokoll in Dialogform
verfasst, teilweise aber auch nur stichpunktartig gehalten. Es war auch nicht
alles ganz genauso, sondern manchmal habe ich erst durch Nachfragen verstanden
worauf sie hinauswollten, aber das ist kein Problem.
Dieses Protokoll wurde in keiner Weise von Professor Vöcking oder Dr. Noll
abgesegnet, spiegelt nur meine Erinnerung wieder und muss nicht dem entsprechen
was die beiden wirklich gesagt haben.


Effiziente Algorithmen (Glück gehabt, genau meine Wunschthemen)

V: Fangen wir am Anfang der Vorlesung an. Was sind den Flußprobleme?
Ich:  Graph, Kapazitätsfunktion, Fluß, Flußerhaltung, Kapazitätsbeschränkung
      (er wollte das meiste gar nicht so genau wissen, aber ich brachte die
      Definitionen teilweise dann später bei Bedarf).

V: Ford-Fulkerson, wonach sucht man da?
Ich:  Flußvergrößernde Wege im Restnetzwerk, Restnetzwerk erklärt.

V: Die Korrektheit zeigt man ja mit dem Mincut-Maxflow Theorem, skizzieren Sie
   das doch mal.
Ich:  Die 3 Teile erklärt und rudimentär aufgeschrieben.

V: Und wie beweist man die einzelnen Folgerungen?
Ich:  (a) nach (b) erklärt, (b) nach (C) erklärt mit Skriptansatz für die
      Aufteilung in die beiden Partitionen eines Cut, aber dann mit Induktion,
      was aber nicht ganz reichte (Wobei er nachher spaßeshalber einen Kollegen
      nannte, der das in seiner Vorlesung auch immer vergessen würde). Was dann
      noch fehlte haben wir ausdiskutiert, ich glaube es war, die Begründung
      warum es keine Rückwärtskanten gibt.

V: Wir hatten diese bessere Lautzeitschranke...
Ich:  Erstmal doch kurz mC erwähnt und dann die n^5 erklärt, aber ohne Beweise.

V: Kommen wir zu Approximationsalgorithmen, da gibt es den Algorithmus von
   Christofides, was macht der so?
Ich:  Eine 3/2-Approximation für das TSP...

V: Was für ein TSP?
Ich:  Das metrische, da muss die Dreiecksungleichung gelten.

V: Kann man das allgemeine TSP approximieren?
Ich:  Nein, was man durch eine Reduktion auf das Hamiltonkreis-Problem zeigen
      kann.

V: Ja, da muss man immer aufpassen wierum man reduziert.
Ich:  Achja andersrum, man reduziert das Hamiltonkreis-Problem auf das TSP.

V: Ja, dann erklären Sie doch mal wie der Christofides arbeitet.
Ich:  Der sucht einen MST (ungerade Grad Kanten, Anzahl, Matching, blahblah)

V: Wie kommen die 3/2 zu Stande?
ICh:  Der MST macht 1 aus, denn wenn man eine optimale Tour betrachtet kann
```



```
      man eine Kannte streichen und dann kriegt man einen MST...

V: (Dreht sich kopfschüttelnd zu Dr. Noll um) Das sagt jeder, also fast jeder,
   aber das stimmt nicht. (Er malt mir das auf).
Ich:  Ok, das ist also nur ein Spanningtree, kein minimaler, aber der minimale
      muss ja kleinergleich sein und deswegen auch kleiner als eine optimale
      Tour und deswegen kann man das mit den Kosten einer optimalen Tour
      abschätzen.

V: Und das Matching?
Ich:  Ja, dass das 1/2 der Kosten einer optimalen Tour ausmacht zeigt man wieder
      über eine optimale Tour...(Aufteilung in 2 matchings, siehe Skript)

V: Ja dann kommen wir noch kurz zu randomisierten Algorithmen, wir hatten da
   welche zum Mincut-Problem.
Ich:  Ja Contract... und Fastcut.

V: Wir ging denn Contract vor?
Ich:  Idee: wenige Kanten im Mincut, kickt uniform zufällig eine raus,
      kontrahiert (usw.)

V: Wie hoch ist denn die Erfolgswahrscheinlichkeit?
Ich:  2/n^2, wobei ich mich eigentlich immer gefragt habe warum wir 2/n^2
      genommen haben und nicht 1/n^2, weil eigentlich war das ja 2/(n*(n-1)),
      aber bei den n haben wir ja auch "verschönert". (Hab dann noch ne halbe
      Minute mit ihm drüber geredet ;)).

V: Die Wahrscheinlichkeit ist ja jetzt noch nicht so toll, wie kann man das
   denn besser machen?
Ich:  Durch Wahrscheinlichkeitsamplifikation, wir hatten da diese vom Himmel
      gefallene Formel (1 - 1/x)^x <= 1/e

V: Ja so ganz vom Himmel gefallen ist die nicht ... die hatten Sie bestimmt in
   der Schule (Vöcking und Noll grinsen, aber ich hab es grinsend abgestritten
   :)).
Ich:  Auf jeden Fall kann man die Erfolgswahrscheinlichkeit jetzt da einsetzen,
      und dann ... Wiederholungen... Gegenwahrscheinlichkeiten ... konstant...
      blah... (Einsetzen vorgeführt).

V: So dann sind wir durch, (guckt zu Dr. Noll) ich hab schon ein wenig
   überzogen...


PaCo

Noll: Was macht man bei PaCo überhaupt, was sind die Ziele?
Ich:    Optimierung von Programmen, bzgl. Laufzeit, Speicherplatzbedarf,
        Codelänge

Noll: Was für Optimierungen hatten wir für SLC?
Ich:    CSE (macht das und das), Konstantenfaltung (macht das und das) und
        ähh... Mist.. (Sein Tipp "Es hat etwas mit Konstanten zu tun" muss
        wohl falsch gewesen sein, sein nächster Tipp "Es hat etwas mit totem
        Code zu tun" war dann eher ein Zaunpfahl ;), aber ich hab dann schön
        erklärt was Dead Code Elimination ist.)

Noll: Bei Live Variable Analyse: wie geht das genau?
Ich:    gen und kill mit leichtem Straucheln wg. Fallunterscheidung erklärt.
        Hab das dann auch nicht mehr 100% hinbekommen, es hakte an einem
        Detail, aber meine (wahre) Ausrede, dass es schwer wäre sich in der
        Prüfung darauf zu konzentrieren hat er wohl durchgehen lassen.

Noll: Bei IC, was hatten wir da so für Optimierungen?
Ich:    öh ... ja also erstmal dieselben wie bei SLC...

Noll: Ja, aber da war das ein wenig anders, wie denn?
Ich:    Da betrachtet man nur Programmpfade statt Berechnungspfade, wir hatten
        da MOP und MFP (und das recht anschaulich erklärt und wo die Ideen und
        Unterschiede liegen). Bei der Darstellung des Gleichungssystems mit den
```

```
        Transferfunktionen phi für MFP erstmal was falsches hingeschrieben, aber
        das nach Hinweis und kurzem Nachdenken dann korrigiert und erklärt.

Noll: Und wie läuft das mit dem Gleichungssystem ab, wenn man die Analyseinfo
      bestimmen will?
Ich:    T_delta erklärt und gesagt, dass man Fixpunktiteration macht bis einer
        gefunden ist.

Noll: Warum funktioniert das?
Ich:    Weil die Analyseinformationen aus einem vollständigen Verband mit ACC
        stammen und die phi monoton und hier damit auch stetig sind und man
        daher den Fixpunktsatz von Tarski anwenden kann der besagt, dass es
        Fixpunkt gibt (noch ein bisschen erklärt, aber ohne Beweise).

Noll: Sind MFP und MOP denn gleich?
Ich:    Nein, nur wenn die Transferfunktion distributiv ist (Formel
        aufgeschrieben).

Noll: Wir hatten da so ein Beispiel in der Vorlesung wo MFP und MOP
      unterschiedlich sind, ist Ihnen das noch parat?
ICh:  Ja da hatten wir... (das mit x<-2;y<-3 vs. x<-3;y<-2 und nachher z<-y+x).

Noll: So dann kommen wir mal zur funktionalen Programmierung


FP (ích fand hier hat er recht oberflächlich geprüft)

Noll: Herr Giesl hatte ja Haskell vorgestellt, was sind denn die Besonderheiten
      der Sprache?
Ich:    Es ist eine funktionale Programmiersprache (kurz Unterschied zu
        imperativ erklärt) und sie hat lazy evaluation (non-strict, left most
        outermost und am wichtigsten: sie behält Referenzen auf Ausdrücke, damit
        diese nicht später mehrfach ausgewertet werden, auch wenn sie vor der
        Auswertung an mehreren Stellen weitergegeben werden.)

Noll: Es wurden 2 verschiedene Arten vorgestellt die Semantik von Haskell
      festzulegen. Welche waren das denn?
Ich:    Denotationelle Semantik mittels mathematischer Definition und
        operationelle Semaktik mittels Reduktionsregeln im Lambda-Kalkül.
        Erklärt, dass es complex und simple Haskell gibt, was die Unterschiede
        sind auch zum richtigen Haskell (Hauptunterschiede: Patternmatching und
        Typklassen). Und dass man complex in simple und simple ggf. auf lambda-
        kalkül reduziert.

Noll: Wie war denn der Domain von Haskell definiert?
Ich:    Konstruktoren mit verschiedenen Anzahlen Parameter, Funktionen, ...
        (Wobei ich zwar recht unsicher war, aber er das auch gar nicht hören
        wollte und mir zustimmte, aber mich abbrach weil er wohl auf was
        anderes hinaus wollte.)

Noll: Wie war denn der Funktionendomain definiert?
Ich:    (Nachdem ich gepeilt hatte, dass er vor allem auf die Halbordnung
        zwischen verschiedenen Funktionen hinaus wollte) f < g wenn für alle
        f(d) != _|_ gilt f(d) = g(d).

Noll: Beim Lambda-Kalkül gab es ja die verschiedenen Reduktionen, davon war die
      beta-Reduktion die wichtigste, was machte die denn?
Ich:    Funktionsapplikation, kleines Beispiel gemacht \x.x 5 ->_beta 5.

Noll: Was ist denn bei (\x.t) u wobei t ein term und u ein Ausdruck ist zu
      beachten?
Ich:     Dass t gebundene Variablen enthalten kann, die von der Bezeichnung her
         mit denen in u übereinstimmen und deswegen vorher umbenannt werden
         müssen. Das geht mit der alpha-Reduktion.

Noll: Danke, dann gehen Sie mal bitte raus und wir beraten uns.

Ich stand dann 3 Minuten ginsend im Flur, weil es bestens gelaufen war und ich
vor der Prüfung nicht so recht wusste, ob es so sein würde. Mit der folgenden
```



```
1,3 war ich dann auch sehr zufrieden.

Eine generelle Einschätzung zum Abschluss noch. Ich glaube Professor Vöcking
fragt recht speziell, leitet das aber jeweils gut ein und möchte dann, dass
man anschaulich erklärt. Formeln sind nicht sooo wichtig, aber wenn man mal
z.B. die Fehlerwahrscheinlich direkt parat hat ist das nicht schlecht. Dr. Noll
prüft etwas leichter und allgemeiner denke ich, aber auch er legt Wert auf
Verständnis und nicht auf das Pauken von Formalien. Er hat mir z.B. vorher per
E-Mail zugesagt, dass man die Transfortmationsregel von complex nach simple
Haskell nicht auswendig wissen muss. Ich habe mich nachher noch kurz mit ihnen
unterhalten und sie sagen selbst, dass sie mehr Wert auf Verständnis legen.
Beide sind überaus freundlich und Professor Vöcking war selbst als ich meinen
ursprünglichen Prüfungstermin eine Woche vorher um ein paar Wochen verschoben
habe sehr freundlich und hat mit mir überlegt wie wir das am besten machen,
weil das mit dem Beginn meines Auslandsjahres in Göteborg sehr knapp wurde.
Ich kann die beiden also nur empfehlen :).

Generell ist übrigens gut, wenn man einfach redet und alles Wissen was grad
irgendwie zur Frage passt und was einem parat ist vorbringt. Das zeigt Ver-
ständnis und füllt die Zeit mit Dingen, die man kann und lässt weniger Raum
und Zeit für allzuviele Rückfragen. Natürlich sollte man schauen, dass man
nicht zu viel abweicht und sich vor allem auf das zubewegt was sie hören
möchten. Und wenn man was nicht versteht oder nicht weiss was gemeint ist
einfach nachfragen.

An dieser Stelle noch ein Hinweis auf etwas von dem ich leider erst zu spät
erfahren habe: In der Fachschaft gibt es einen sogenannten "digitalen Schrank",
der wohl unter anderem auch 3 Protokolle zu FP-Prüfungen von Pr. Giesl enthält,
wovon man im gedruckten Ordner keine findet. Darüber hinaus habe ich auch noch
Protokolle von Vöcking als er noch in Dortmund war, die ein Studienkollege
wohl über google aufgespürt hat. Einige gibts nicht auf s-inf.de oder im Ordner
in der Fachschaft, vielleicht aber im digitalen Schrank. Ich sende sie aber
auch gerne per E-Mail, wenn sie jemand haben möchte.

Ich wünsche euch viel Erfolg bei der Prüfung!!!
```