Gedächtnisprotokoll Diplomprüfung Theoretische Informatik

Professores Jürgen Giesl und Juraj Hromkovic

Stefan Buhr
Email: `stefan.buhr@rwth-aachen.de`

6. Mai 2005

Zusammenfassung

Dieses Prüfungsprotokoll versucht einige Themen zusammenzutragen, die im Laufe meiner Prüfung von den beiden genannten Professores Giesl und Hromkovic angeschnitten wurden. Grundsätzlich kann ich keine Garantie auf Vollständigkeit geben. Hier stichpunktartig eine kurze Übersicht über die Prüfung:

Datum der Prüfung: 6. Mai 2005, 11.45 Uhr
Prüfer: Professor Jürgen Giesl, Lehr- und Forschungsgebiet 2 für Informatik, sowie Professor Juraj Hromkovic, ETH Zürich
Protokollführung: jeweils abwechselnd die beiden Prüfer
Geprüfte Fächer:

- Effiziente Algorithmen
  (Buch „*Algorithmics for hard problems*“, 2. Auflage)
- Termersetzungssysteme (Vorlesung aus dem SS 2004)
- Logikprogrammierung (Vorlesung aus dem WS 2003/04)

Dauer: ca. 50 Minuten
Prüfungsnote: 1.0

1 Effiziente Algorithmen

Parametrisierte Komplexität

- Erzählen Sie mir doch einmal etwas über Parametrisierte Komplexität!

  Funktion „Par" definiert und erläutert (inkl. $Set_U(k)$ unendlich für unendlich viele k). Hier kam von Professor Hromkovic die Zwischenfrage, warum wir die Unendlichkeit von $Set_U(k)$ fordern. Definition von Par-parametrisierten Polynomzeit-Algorithmen geliefert.

- Welche *Par*-parametrisierten Algorithmen kennen Sie denn?

  Zwei Algorithmen für das Vertex-Cover-Problem. Wollen Sie einen davon sehen?

- Beide. *Algorithmen erklärt und Komplexitätsabschätzung geliefert.*

Diamond-Graphen

- In welchem Zusammenhang haben wir Diamanten kennengelernt?

  Wir haben uns einen Graphen aus acht Knoten gestrickt, der wie folgt aussieht(aufgezeichnet). Anschaulich erläutert, dass diese Graphen für einen Hamiltonkreis nur entweder in Nord-Süd- oder West-Ost-Richtung durchlaufen werden können.

- Genau. Das haben wir für zwei Dinge gebraucht? Wissen Sie für welche?

  Reduktion $HC \leq_P RHC$. Erwähnt, dass diese Reduktion Bestandteil des Nachweises war, dass es keine „exact, polynomial-time searchable neighborhood" für TSP gibt. Reduktion aufgezeichnet und dargelegt. Im Anschluß wesentliche Merkmale des pathologischen TSP-Falls, im Zusammenhang mit dem die Diamond-Graphen auch zur Anwendung kamen, genannt und erläutert.

Schöning's Algorithmus

- Kennen Sie Schöning's Algorithmus?

Ausführlich die Vorgehensweise des Algorithmus erläutert.

- Welche Wahrscheinlichkeiten sind da involviert?

  Genauer auf die Bedeutung der Wahrscheinlichkeiten p_j, q_{ij}, q_j, p, $\frac{1}{3}$, $\frac{2}{3}$ *eingegangen, teilweise graphisch illustriert.*

2 Termersetzungssysteme

- Schreiben Sie doch einmal ein Gleichungssystem für die Konkatenation zweier Listen auf!

  ⟨UFF!⟩ Das könnte ich direkt als Logikprogramm aufschreiben.

- Dann machen Sie das doch zunächst einmal. Das recyclen wir dann gleich für den Teil „Logikprogrammierung"!

  Einfaches append-*Logikprogramm aufgeschrieben. In der Zwischenzeit krampfhaft darüber nachgedacht, wie ich das am Sinnvollsten als Gleichungssystem aufschreibe. Ich wusste, dass wir das irgendwo schon einmal hatten. Nach der Prüfung fiel mir auf, dass wir das Gleichungssystem in der Scheinklausur, 1. Aufgabe hatten. Ist mit leider in der Prüfung nur mit Hilfe von Professor Giesl gelungen:*

$$\begin{aligned} append(nil, z) &\equiv z \\ append(cons(x, y), z) &\equiv cons(x, append(y, z)) \end{aligned}$$

- Schreiben Sie mal eine Termgleichung, die behauptet, dass das neutrale Element nicht nur links-, sondern auch rechtsassoziativ ist.

$$append(x, nil) \equiv x.$$

- Und was macht man jetzt damit?

  Wir wollen prüfen, ob die Gleichung aus der Gleichungsmenge folgt, d. h. ob jedes Modell der Gleichungsmenge auch eines der Gleichung ist. Dazu basteln wir uns zunächst einmal durch Richten der Gleichungen ein äquivalentes Termersetzungssystem.

- Ok, durch Richten der Gleichungen erhält man ein *äquivalentes* Termersetzungssystem. Welche Eigenschaften müssen wir denn noch fordern?

  Wir müssen das TES auf Konvergenz prüfen, d. h. ob es terminiert und konfluent ist.

- Und? Terminiert das Gleichungssystem?

  Ja, nach dem Satz von Manna und Ness muss ich ja nur prüfen, ob eine Reduktionsrelation existiert, so dass $l \succ r$ für alle $l \to r \in \mathcal{R}$ existiert. Und in diesem Beispiel kann man das schnell mit LPO zeigen (kurz vorgeführt, d. h. erste Regel klappt schon mit der Einbettungsordnung und mit der Präzendenzwahl „append $\sqsupset$ cons“ funktioniert das auch bei der zweiten Regel).

- Wo ist hier jetzt der lexikographische Vergleich?

  Grundsätzlich Idee des lexikographischen Vergleichs erklärt. Erst später ist mir aufgefallen, dass hier zunächst gar kein lexikographischer Vergleich notwendig ist, da die beiden führenden Funktionssymbole unterschiedlich sind...

- Was ist die lexikographische Pfadordnung eigentlich?

  Eine Simplifikationsordnung.

- Warum benutzen wir denn eigentlich Simplifikationsordnungen, wenn nach dem Satz von Manna und Ness nur eine Reduktionsrelation gefordert wird?

  Simplifikationsordnungen haben den großen Vorteil, dass man hier „nur“ die Eigenschaften Stabilität, Monotonie, Transitivität, Irreflexivität und die Teiltermeigenschaft nachweisen braucht. Das gibt der Satz von Kruskal her.

- Aber das sind doch relativ viele Eigenschaften, die wir hier zeigen müssen. Wo steckt denn der große Vorteil?

  Dass wir die Fundiertheit nicht separat zeigen müssen.

- Können Sie das zeigen?

  Satz von Kruskal, Teil c) bewiesen unter Angabe und Einbeziehung der Resultate der beiden vorangegangenen Teile.

- Ok, was müssen wir denn dann bei unserem TES prüfen?

  Wir müssen das TES auf Konfluenz prüfen. Hierzu bildet man alle kritischen Paare und prüft, ob diese zusammenführbar sind.

- Und? Wie ist das in unserem Termersetzungssystem?

  Da gibt's keine kritischen Paare. Also ist das bereits in der Form konfluent.

- Und? Folgt die Gleichung aus der Gleichungsmenge?

  Nein, ich denke nicht.

- Das ist aber doch komisch. Immerhin sprechen wir doch hier von Listen. Da ist das doch so...

  Das liegt daran, dass man die Funktionssymbole und Konstanten interpretieren darf, wie man will. Und es gibt halt Algebren, in denen die Gleichung nicht aus der Gleichungsmenge folgt. Wir hatten da so was Pathologisches für das plus-TES auf der Vorlesung, mit $\Diamond$- und $\Box$-Symbolen.

- (An dieser Stelle habe ich nicht recht verstanden, worauf Professor Giesl in den folgenden Fragen hinaus wollte. Aber er wollte irgendwie wissen, worin das Problem liegt, wenn man so künstliche Symbole in den Träger einschleust.)

- Wie kann man denn jetzt nachweisen, dass die Gleichung wenigstens induktiv gültig ist?

  Konsistenzbeweismethode und Konsistenzbeweismethode für konvergente TES erklärt. Kurz auf die Regeln des Vervollständigungsalgorithmus' hingewiesen, die im Konsistenzbeweisverfahren „recycled" werden.

- Welche Möglichkeiten gibt es denn für die Ergebnisse des Konsistenzbeweisverfahrens?

  Naja, die Vervollständigung kann zum einen nicht terminieren. D. h. ich kann immer wieder Gleichungen richten, weil die linke Seite bezüglich der eingegebenen Reduktionsrelation größer als die rechte Seite ist, aber in jedem Schritt kommt wenigstens eine neue Gleichung hinzu. Möglicherweise kann ich auch irgendwann eine Gleichung nicht mehr *richten. Die wird dann zu einer persistenten Gleichung und die Vervollständigung schlägt fehl. Man könnte den Algorithmus dann noch einmal mit einer anderen Reduktionsrelation neu starten. Und möglicherweise findet die Regel „Inkonsistenz" Anwendung, d. h. ich finde zwei Grundnormalformen, die durch den Algorithmus gleich gemacht werden.*

- Eine wichtige Möglichkeit haben Sie noch vergessen...

  Oh, natürlich. Es kann natürlich auch vorkommen, dass alle kritischen Paare der persistenten Regeln einmal betrachtet wurden und keine Regel mehr anwendbar ist. Dann kann man die induktive Gültigkeit der Gleichung folgern.

- Und? Ist unsere Gleichung jetzt wenigstens induktiv gültig?

  Kurz nachgedacht, etwas erklärt und Gleichung für induktiv gültig befunden.

3 Logikprogrammierung

- Wir haben ja bereits das Logikprogramm für *append* aufgeschrieben:

  $$append(nil, Xs, Xs).$$
  $$append([X|Xs], Ys, [X|Zs]) :- append(Xs, Ys, Zs).$$

  Was passiert denn beispielsweise bei der Anfrage $append(X, Y, Z)$?

  Prolog-Auswertungsstrategie erläutert. Dabei unter anderem darauf eingegangen, warum es sinnvoll ist, die Fakten zuerst einzugeben.

- Ist der SLD-Baum zu dieser Anfrage denn endlich oder unendlich?

  Der ist wohl unendlich, weil die entstehende Zielklausel immer wieder mit der zweiten Regel unifiziert.

- Welche Nichtdeterminismen können denn grundsätzlich in der Logikprogrammierung auftreten?

  Nichtdeterminismen 1. und 2 .Art erläutert.

- Wie sieht es denn mit den Nichtdeterminismen aus? Sind die kritisch?

Vertauschungslemma erwähnt, welches grundsätzlich besagt, dass zwei aufeinanderfolgende SLD-Resolutionsschritte problemlos vertauscht werden können.

- Ja, aber ganz so einfach ist das ja nicht. Ist es immer egal, mit welchem Literal resolviert wird?

  Hmmmm, nein. [Da war doch was im Prüfungsprotokoll von Matthias Sondermann... ;-)] Betrachten wir mal folgendes Logikprogramm:

  $$p :- p.$$

  mit der Anfrage $?- p, q$. *Wählt Prolog stets das erste Literal der Zielklausel zur Resolution, entsteht eine unendliche Berechnungsfolge. Wählt Prolog hingegen das zwei Literal und stellt fest, dass* q *nicht beweisbar ist, so bricht das Programm vorzeitig ab, terminiert also.*

- Dann werden in der Logikprogrammierung verschiedene Semantiken eines Logikprogramms betrachtet. Welche kennen Sie?

  Die deklarative, die prozedurale und die Fixpunkt-Semantik.

- Was ist denn Fixpunktsemantik?

  Funktion $\mathbf{trans}_\pi : 2^{\mathbf{Prim}_\Sigma} \to 2^{\mathbf{Prim}_\Sigma}$ *erläutert.* $F_P[\pi, G]$ *erklärt. Erwähnt, dass es sich bei* $(\mathbf{Prim}_\Sigma, \subseteq)$ *um einen vollständigen Verband handelt, und die Funktion* $\mathbf{trans}_\pi$ *monoton und stetig ist. Mit dem Satz von Tarski und Knaster hat diese Funktion dann einen eindeutigen kleinsten Fixpunkt, der sich durch iterierte Anwendung der Funktion auf das kleinste Element des vollständigen Verbandes (hier* $\emptyset$*) berechnen lässt.*

- Ich wollte Sie gerade fragen, wo jetzt genau der Fixpunkt da ist. Aber das haben Sie ja jetzt schon vorweggenommen. Gut, das reicht dann glaube ich auch.

Fazit

Als ich die drei Prüfungsfächer fixiert hatte, und angefangen habe zu lernen, hatte ich keine Ahnung, wie umfangreich der Stoff zu Effiziente Algorithmen ist! Habe mir irgendwann in der Lernphase im LuFG I2 das Skript zu „Automatisierte Programmverifikation“ ausgeliehen und kopiert und bin mittlerweile der festen Überzeugung, dass das in Kombination mit TES die einfachere Option gewesen wäre, da es dort nicht unerhebliche stoffliche Überschneidungen gibt! Ich will mir nur nicht sagen lassen, ich hätte Euch nicht gewarnt... ;-)