Prüfungsprotokoll zur Diplomprüfung

Theoretische Informatik

Prüfungsgebiete:

- Compilerbau (Indermark, SS 04)
- Logikprogrammierung (Indermark, WS 03/04)
- Termersetzungssysteme (Giesl, SS 04)

Prüfer:	Prof. Dr. Klaus Indermark
Beisitzer:	Dr. Benedikt Bollig
Datum:	31.08.2005 (11:00 Uhr)
Dauer:	knapp über 45 min.
Note:	1.3

Prof. Indermark fing nach einer kurzen freundlichen Begrüßung direkt mit der Prüfung an. Er fragte nur kurz, ob die Reihenfolge CB, LP, TES erwünscht sei und legte los.

1. Compilerbau (15 min)

   - Berechnung der LR(0)-Mengen durch Potenzmengenkonstruktion von NFAs. (Anfangszustand angeben und erklären, welche zwei Arten von Transitionen es gibt)
   - Wann gibt es Reduce-Reduce-Konflikte? (Meine Antwort war hier zunächst nicht zufriedenstellend. Daher kam Prof. Indermark zur folgenden Nachfrage.)
   - Wie sind denn die LR(0)-Auskünfte definiert? (Aus $S \Rightarrow_r^* \alpha A w \Rightarrow_r \alpha\beta_1 \cdot \beta_2 w$ folgt $[A \to \beta_1 \cdot \beta_2] \in LR(0)(\alpha\beta_1)$. Er wollte wohl schließlich hören, dass bei zwei Reduce-Auskünften $[A \to \alpha\cdot]$ und $[B \to \beta\cdot]$ in der selben LR(0)-Information α ein Suffix von β ist oder umgekehrt bzw. er wollte sogar diese Aussage verallgemeinert auf das obige $\alpha\beta_1$ haben.)
   - Shift-Reduce-Konflikte bei LR(1)-Grammatiken erklären. (Ihm kam es nur darauf an, dass das look-Ahead-Symbol bei der Shift-Auskunft direkt hinter dem Punkt steht und eben nicht in der zweiten Komponente, wie bei einer Reduce-Auskunft.)
   - Attributgrammatiken: Wenn man einen attributierten Ableitungsbaum mit aktuellen Attributvariablen hat, wie werden dann die Attributvariablen berechnet? (Hab ihm zunächst erzählt, dass Attribute in inherite und synthetische zerfallen und welche von welchen abhängen dürfen (Innenvariablen von Außenvariablen). Wichtig war ihm, wie die formalen Attributvariablen der Regeln durch Indexsubstitution zu aktuellen Attributvariablen im Baum verklebt werden.)

- Unterschied zwischen strikter und und nicht-strikter Auswertung boolescher Ausdrücke und Erzeugung von Jumping Code. (Es hat Prof. Indermark hier nicht gereicht, dass ich ihm das qualitativ erläutern konnte, sondern er wollte die Übersetzung anhand eines Beispiels schon konkreter wissen, d. h. ich sollte folgendes hinschreiben:

  $$\mathrm{nbt}(E_1 \text{ AND } E_2, st, a, a_t, a_f, l) := \mathrm{nbt}(E_1, st, a, a', a_f, l)\ \mathrm{nbt}(E_2, st, a', a_t, a_f, l)$$

  Das musste ich mir während der Prüfung erst zurechtbiegen. Danach war wahrscheinlich auch klar, dass er mich höchstens noch auf 1.3 prüft.)

2. Logikprogrammierung (15 min)

   - Was ist ein Logikprogramm? (eine endliche Menge endlicher definiter Hornklauseln)
   - Was ist eine Hornklausel?
   - Was ist ein Literal? (positives oder negatives Prädikat $p(t_1, \ldots, t_n)$, die t_i sind Terme)
   - Wie kann man eine Klauselmenge als Formel schreiben? (Literale werden durch $\vee$ verknüpft, Klauseln werden durch $\wedge$ verknüpft und Variablen sind allquantifiziert.)
   - Was bedeutet die Folgerungsbeziehung $\pi \models G$? (Allgemein bedeutet das in der Logik, dass alle Interpretationen $\mathcal{I} = (A, \alpha, \beta)$, die alle Formeln aus π erfüllen, auch Modell von G sind.)
   - Was ist eine Interpretation? Was bedeuten diese A, α, β? (besonders die Deutung α will er ganz formal haben)
   - Warum haben wir in der Logikprogrammierung keine Variablenbelegung β? (wir betrachten nur Formeln in Skolemnormalform)
   - Wir beschränken uns ja auf Herbrandstrukturen. Was sind das? (Träger besteht nur aus Grundtermen, aufgebaut aus den Konstanten- und Funktionssymbolen. Formal hinschreiben, wie Funktionssymbole auf natürliche Weise gedeutet werden.)
   - Was ist die Herbrandstruktur einer Struktur?
   - Warum sind in $p_{\mathcal{T}_\mathcal{A}}(t_1, \ldots, t_n) := p_\mathcal{A}([\![t_1]\!]_\mathcal{A}, \ldots [\![t_n]\!]_\mathcal{A})$ die $[\![t_i]\!]_\mathcal{A}$ überhaupt wohldefiniert? (Prof. Indermark hat nach kurzem zögern sofort die Antwort verraten: Das ist trivial, da der Träger $\mathcal{T}_\mathcal{A}$ ja gerade die Grundterme von $\mathcal{A}$ sind.)
   - Deklarative Semantik? (also Definition von $D[\![\pi, G]\!]$ aufschreiben)
   - Praktischer Teil zur LP: Programm zum Invertieren von Listen, also das *reverse*, aufschreiben (mit und ohne Akkumulatortechnik).

- Definite Clause Grammars (DCGs). Hab das in folgender Reihenfolge erklärt (Prof. Indermark hat natürlich sehr kleinschrittig nachgefragt): Die Schreibweise von Regeln in Prolog, Bedeutung von Listen und Prädikaten, Bedeutung einer Regel $a(W)$, Umsetzung einer solchen Regel in DCG-Notation in Prolog (er wollte hier zunächst nur den einfachen Fall mit *append*, also ohne Differenzlisten).
- DCGs mit Differenzlisten. Musste ich nicht ausformulieren. Ich sollte ihm nur die Idee erklären. Wichtig war, dass ein Prädikat $a(W, R)$ beweisbar ist, wenn ein Wort W abzüglich eines Suffixes R aus einem Nichtterminal A ableitbar ist.

3. Termersetzungssysteme (15 min)

- Prof. Indermark hat gleich mal zwei Begriffe in den Raum geworfen, die mir nichts sagten, da sie Prof. Giesl nie verwendet hat (hab diese Begriffe auch schon wieder vergessen). Das hab ich ihm auch gleich gesagt. Bezeichnen sollten diese Begriffe wohl die beiden Relationen $\equiv_{\mathcal{E}}$ und $\leftrightarrow_{\mathcal{E}}$.
- Was bedeutet denn $s \equiv_{\mathcal{E}} t$? ($\mathcal{E} \models s \equiv t$, also alle Algebren (nicht Interpretationen wie in der Logikprogrammierung), die alle Gleichungen aus dem Gleichungssystem E erfüllen, erfüllen auch die Gleichung $s \equiv t$.)
- Was bedeutet das, wenn eine Algebra eine Gleichung erfüllt? (Alle Interpretationen mit allen möglichen Variablenbelegungen erfüllen diese Gleichung.)
- Wozu ist das Äquivalent? (Zur Beweisrelation $\leftrightarrow_{\mathcal{E}}$, das ist die symmetrische Hülle der Ersetzungsrelation. Wie $\rightarrow_{\mathcal{E}}$ definiert ist, sollte ich ihm nicht erklären. Er meinte, ich wüsste das eh.)
- Welche Richtung des Äquivalentsbeweises ist denn einfach? (von $\leftrightarrow_{\mathcal{E}}$ nach $\equiv_{\mathcal{E}}$)
- Und wie nennt man diese Richtung? (Ich meinte zunächst, dass das ja Ansichtssache ist, wo er mir nicht ganz Unrecht gegeben hat. Aber diese Richtung nennt man Korrektheit und nicht Vollständigkeit, da man ja die semantischen Beziehung durch die syntaktische Beziehung ausdrücken will und nicht umgekehrt.)
- Wann kann man denn $s \equiv_{\mathcal{E}} t$ zeigen? (Allgemein ist das Problem unentscheidbar. Aber wenn man z. B. nur Grundidentitäten hat oder aber ein zu $\mathcal{E}$ äquivalentes und konvergentes TES finden kann, ist das Wortproblem entscheidbar.)
- Was bedeutet konvergent? (konfluent und terminierend)
- Was bedeutet Konfluenz? (Definition angegeben und gesagt, dass Indeterminismen zusammenführbar sind)
- Wie löst man dann das Wortproblem $s \equiv_{\mathcal{E}} t$?

- Welche Eigenschaft braucht man noch, um zu zeigen, dass dieser Algorithmus korrekt ist? (Prof. Indermark hat diese Frage irgendwie so langatmig formuliert, dass ich sie nicht verstanden hab. Es war mir aber klar, dass er nun auf die Church-Rosser Eigenschaft hinaus wollte.)
- Beweis der Äquivalenz von Church-Rosser Eigenschaft und Konfluenz.
- Wie überprüft man, ob ein TES konfluent ist? (Ich wollte zunächst direkt was von kritischen Paaren erzählen, aber Prof. Indermark wollte natürlich erstmal auf die lokale Konfluenz kommen.)
- Wann ist lokale Konfluenz zur Konfluenz äquivalent? (Terminierung)
- Welcher Satz zeigt das? (Diamond Lemma bzw. Satz von Newman)
- Welche Situationen gibt es da? Zeichnen sie doch mal so einen Term als Baum. (Prof. Indermark wollte hier zwar wirklich nur die Zeichnungen, aber dafür sollten die dann auch wohl einigermaßen ordentlich sein. Insbesondere wollte er, dass ich nicht einfach einen Teilbaum $l_1\sigma_1$ male, sondern die Substitution σ_1 unterhalb von l_1 extra einzeichne. Auch hatte ich aus Schusseligkeit den Baum zunächst unten offen gelassen, was er natürlich sofort kritisierte. Ich durfte auch nur noch den einfachsten Fall erklären (Stellen nebeneinander), dann hat Prof. Indermark die Prüfung beendet.)

Fazit: Prof. Indermark ist insgesamt ein sehr empfehlenswerter Prüfer, wenn man eine 1.3 als Note anpeilt. Man kann wunderbar nach den Prüfungsprotokollen lernen, da er anscheinend nur die von den Prüfungsprotokollen abgedecktem Themen anspricht. Es kommt ihm bei seinen Fragen in der Regel nur auf ein bestimmten Punkt an, oder aber er will nur wissen, ob man die Sache verstanden hat. Er geht dann sofort zur nächsten Frage über, wenn dieser Punkt genannt wurde oder er merkt, dass man sich bei dem Thema sehr sicher auskennt. Man redet daher bei der Prüfung nie längere Zeit am Stück. Allerdings will er auch sehr oft Dinge sehr genau wissen und hakt konsequent sehr detailliert nach bis man den Punkt gesagt hat, den er hören wollte. Transferwissen über den Vorlesungsstoff hinaus hat er bei mir gar nicht abgefragt.
Prof. Indermark hat mir vorgeworfen, dass ich zunächst immer nur sehr schwammig geantwortet hätte. Ich habe mich quasi immer vorsichtig an seine Fragen herangetastet. Insbesondere wusste ich auch nicht immer sofort, worauf er hinaus wollte. Auch sollte man sich allgemein beim Beantworten der Fragen Mühe geben, auch wenn Prof. Indermark oft sehr schnell zur nächsten Frage übergeht. Mir fiel das sehr schwer und ich tendierte sicherlich zu schusseligen Erklärungen, weil die Fragen oft so einfach und grundlegend waren.
Die 1.3 ist Indermarks Standardnote, mit der ich auch sehr zufrieden bin. Dafür ist es meiner Meinung nach ausreichend, den Stoff gut zu beherrschen und die Prüfungsprotokolle zu kennen. Die Definitionen zu den Fragen sollte man unbedingt drauf haben. Wer eine 1.0 haben will, sollte aber darüber hinaus vielleicht eher folgende Vorgehensweise versuchen: Zunächst genau über Prof. Indermarks frage nachdenken und dann versuchen, eine sehr präzise Antwort zu geben. Das ist natürlich sicherlich nicht ganz so einfach. Wie bereits erwähnt, sollte man sich beim Beantworten der Fragen Mühe geben und möglichst präzise bleiben.

Zum Lernen verwendete Literatur:

- Compilerbau und Logikprogrammierung Mitschriften (gibt es auf www.s-inf.de), offizielles TES Skript vom LuFG i2 (gibt es dort im Sekretariat als Kopiervorlage), zusätzlich TES Mitschrift (gibt es auf www.s-inf.de)
- Videoaufzeichnungen (www.s-inf.de)
- Prüfungsprotokolle (sind fast alle auf www.s-inf.de, aber es lohnt sich auch mal in der Fachschaft vorbeizuschauen)
- Zusammenfassungen von Stefan Hoferer auf `http://www.shoferer.de`. Diese Zusammenstellungen von Prüfungsfragen und Definitionen haben mir viel Lernaufwand abgenommen. Allerdings sind diese Zusammenstellungen mit Vorsicht zu genießen, da dort einige schwerwiegende Fehler drinstecken. Aber diese erkennt man schnell, wenn man diese Zusammenstellungen hinterfragt.

Wünsche Euch viel Erfolg bei Euren Prüfungen!