Gedächtnisprotokoll
Theorieprüfung

Prüfer: Professor Grädel, Professor Giesl
Datum: 14.03.2007
Vorlesungen: Komplexitätstheorie, Termersetzungssysteme, Compilerbau
Dauer: 45 Minuten
Note: 1,0

Dies ist ein Gedächtnisprotokoll. Es werden sicher einige Fragen fehlen, Antworten nicht ganz korrekt wiedergegeben sein und die Reihenfolge muss auch nicht unbedingt richtig sein.

Professor Giesl: Welche Reihenfolge möchten Sie denn?
Ich: Komplexitätstheorie, TES und dann Compilerbau.
Professor Grädel: Was bedeutet es, wenn ein Problem vollständig für eine Komplexitätsklasse ist?
Ich: Das Problem ist in der Klasse enthalten und alle Probleme der Klasse lassen sich mit einer geeigneten Reduktion auf dieses Problem reduzieren.
Gr.: Wie ist denn eine Reduktion definiert?
Ich: Für $A \subseteq \Sigma^*$, $B \subseteq \Gamma^*$ gilt $A \leq B$ genau dann, wenn es eine Funktion $f : \Sigma^* \to \Gamma^*$ gibt mit $x \in A \Leftrightarrow f(x) \in B$ für alle $x \in \Sigma^*$.
Gr.: Welche Anforderungen stellt man denn an Reduktionen?
Ich: Sie sollen transitiv sein und um Vollständigkeitsresultate zu zeigen, sollten sie schwächer als die betrachteten Klassen sein.
Gr.: Ja, nehmen wir mal P. Ist eine $\leq_p$-Reduktion zwischen Problemen aus P sinnvoll?
Ich: Nein, dann könnte man das Problem in der Reduktion lösen und dann je nach Antwort auf zwei Standardausgaben abbilden. Dann würde die ganze Komplexität in der Reduktion stecken.
Gr.: Ja, die Reduktionen müssen signifikant schwächer sein. Zeigen Sie mal, dass $\leq_{log}$-Reduktionen transitiv sind.
Ich: Bei $\leq_{log}$-Reduktionen kann man nicht einfach erst die erste Funktion berechnen und danach mit deren Ergebnis die zweite, dabei könnte man nämlich zu lang werden. Also berechnet man die erste nur bei Bedarf.
Gr.: Erläutern Sie das mal genauer.
Ich: (Nicht mehr sicher, wie es genau funktionierte, hab erst mal eine Skizze für $f \circ g$ gemalt) Man lässt g laufen und wenn man ein neues Zeichen von $f(x)$ bracht, dann lässt man f laufen.
Gr.: Ja. Dann schreiben Sie mal die Komplexitätsklassen zwischen P und $PSPACE$ auf.
Ich: $P \subseteq NP \subseteq PH \subseteq PSPACE$.
Gr.: Nennen Sie mal vollständige Probleme für alle Klassen.
Ich: $SAT - HORN$ für P, das folgt direkt aus dem Beweis des Satzes von

Cook, SAT für NP, PH hat keine vollständigen Probleme...
Gr.: ..Ja?
Ich: PH hat vermutlich keine vollständigen Probleme, sonst würde die Hierarchie kollabieren.
Gr.: Ja, wenn sie das beweisen würden, wäre $P \neq NP$. Warum kollabiert PH denn?
Ich: Weil das vollständige Problem dann in einer Stufe wäre und man alle Probleme auf dieses reduzieren kann. Weil sie einzelnen Stufen unter Reduktionen abgeschlossen sind, folgt dann $PH = \Sigma_k^p$ für dieses k.
Gr.: Ok, und $PSPACE$?
Ich: QBF.
Gr.: Dann kann man ja auch die alternierenden Klassen mit diesen Klassen in Verbindung setzen.
Ich: Ja, es gilt $ALOGSPACE = PTIME \subseteq APTIME = PSPACE \subseteq APSPACE = EXPTIME \subseteq AEXPTIME = EXSPACE$, zusätzlich stimmen die Stufen von PH mit den Maschinen überein, die nur eine konstante Anzahl von Alternationen machen.
Gr.: Zeigen Sie mal den Beweis von $P \subseteq ALOGSPACE$.
Ich: (ein Rechteck hingemalt). Eine Berechung ist eine Matrix der Größe $p(n) \times p(n)$. Die Tiefe ist durch die Zeitbeschränkung gegeben und die Breite, da man in dieser Zeit nicht mehr als $p(n)$ Felder besuchen kann. Die Idee ist, dass der existenzielle Spieler unten einen akzeptierenden Endzustand und einen Buchstaben rät. Dann rät er immer die drei Buchstaben die direkt darüber in der Matrix stehen und der universelle Spieler wählt einen dieser Buchstaben. So spielt man sich dann nach oben zur Startkonfiguration und vergleicht dann die geratenen Symbole mit den tatsächlichen.
Gr.: Ok, dass man das in $LOGSPACE$ machen kann ist klar. Warum ist es korrekt.
Ich: Wenn die Maschine nicht akzeptiert, dann muss der existenzielle Spieler in jedem Zug lügen. Dann hat aber der universelle Spieler eine Gewinnstrategie und umgekehrt induziert jede akzeptierende Berechnung eine Gewinnstrategie für den existenziellen Spieler.
Gr.: Ok, Sie haben oben kein nicht-deterministisches Äquivalent zu $PSPACE$ angegeben.
Ich: Es gilt $PSPACE = NPSPACE$ nach dem Satz von Savitch.
Gr.: Was sagt der denn?
Ich: $NSPACE(S) \subseteq DSPACE(S^2)$ für $S(n) \geq log(n)$.
Gr.: Beweisen Sie das mal.
Ich: Dafür definiert man sich die Funktionen $Reach(C_1, C_2, k)$. Die liefert 1 genau dann, wenn $C_1 \vdash^{\leq 2^k} C_2$ gilt (hab den Fall $k > 0$ aufgeschrieben). Da die Maschine nach $2^{O(S(n))}$ Schritten stoppen muss und $Reach$ einen Platzverbrauch von $S(n)$ hat, kommen wir dann bei $O(S^2(n))$ raus.
Gr.: Dann liegen ja zwischen P und $PSPACE$ die probabilistischen Klassen. Erläutern Sie die mal.
Ich: Soll ich das Bild malen oder die Definitionen geben?
Gr.: Zuerst das Bild.

Ich: (Das Inklusionsdiagram gemalt) P ist eine probabilistische Klasse, wenn man die Zufallsbits einfach ignoriert, ist enthalten in ZPP. In dieser Klasse kriegt man nur richtige Antworten, aber mit einer kleinen Wahrscheinlichkeit keine Antwort. ZPP liegt nach Definition im Schnitt von RP und $co-RP$, die liegen beide nach Definition in BPP. Dann kommt PP und alle diese Klassen sind in $PSPACE$, da man die probabilistischen Maschinen simulieren kann und die Ergebnisse aufaddiert. Zusätzlich gilt $RP \subseteq NP$, $co-RP \subseteq co-NP$ und man kann BPP in die zweite Stufe von PH bringen.
Gr.: Definieren sie mal BPP.
Ich: $A \in BPP$ genau dann, wenn es eine PTM M gibt mit $x \in A \Rightarrow PR[M \text{ akzeptiert } x] \geq \frac{2}{3}$ und $x \notin A \Rightarrow PR[M \text{ akzeptiert } x] \leq \frac{1}{3}$ wobei die Konstanten willkürlich gewählt werden können, solange sie nicht zu eng zusammenliegen.
Gr.: Wie begründet man dass?
Ich: Man kann die Irrtumswahrscheinlichkeit beliebig klein kriegen.
Gr.: Wie viel genau?
Ich: Exponentiell kleiner mit polynomial vielen Wiederholungen.
Gr.: Welche Klassen sind denn noch praktisch zu gebrauchen?
Ich: Alles bis BPP.
Gr.: Und warum PP nicht.
Ich: (Fragend) Weil man die Anforderungen schon durch einen Münzwurf hinbekommt? (Nach einem Hinweis von Professor Grädel) Die Schranken liegen zu nah zusammen, man kann die Wahrscheinlichkeiten nicht verbessern.
Gr.: Ja, ist es dieselbe Situation wie bei NP, da kann man die Wahrscheinlichkeiten auch nicht amplifizieren. Dann war es das mit Komplexitätstheorie.

Professor Giesl: Kommen wir zu TES. Welches Ziel hatten wir denn?
Ich: Das Wortproblem zu lösen.
Gi: Und wie ist das definiert?
Ich: Wir haben eine Menge von Gleichungen $\mathcal{E}$ und eine Gleichung $s \equiv t$ und wollen wissen ob $\mathcal{E} \models s \equiv t$.
Gi: Und wie ist das wieder definiert?
Ich: $\mathcal{E} \models s \equiv t$ gilt, wenn alle Algebren (A, α), die Modell von $\mathcal{E}$ sind, auch Modell von $s \equiv t$ sind. Bei dieser Semantik geht man dann über alle Variableninterpretationen.
Gi: Ok, ist das entscheidbar?
Ich: Nein.
Gi: Für welchen Fall ist es denn entscheidbar?
Ich: Wenn wir nur Grundidentitäten betrachten.
Gi: Und im allgemeinen Fall?
Ich: Da ist es semi-entscheidbar.
Gi: Warum.
Ich: Da benutzt man Birkhoff und baut mit der Beweisrelation den Suchbaum von s aus auf. In dem ist t enthalten, wenn die Gleichung gilt. Der Baum kann aber unendlich breit und tief sein.
Gi: Wann ist er unendlich breit?
Ich: Wenn $\mathcal{V}(s) \neq \mathcal{V}(t)$ gilt. Dann muss man diagonal suchen.
Gi: Ja, dann ist Breitensuche schlecht. Schreiben Sie mal ein Gleichungssystem

für die Konkatenation von Listen auf.
Ich: (Schreibe $app(nil, x) \equiv x$ und $app(cons(x, y), z) \equiv cons(x, app(y, z))$ auf.)
Gi: Und noch eine Anfrage, ob die Konkatenation assoziativ ist. Folgt das aus der Gleichungsmenge?
Ich: (schreibe $app(x, app(y, z)) \equiv app(app(x, y), z))$ Nein.
Gi: Warum, normalerweise gilt das doch?
Ich: Weil man auch Algebren hat, deren Träger Elemente enthält, die nicht von Grundtermen getroffen werden. Die kann man dann so hinbiegen, dass sie die Gleichungen aus $\mathcal{E}$ erfüllen, die Anfrage aber nicht.
Gi: Ja, und wie zeigt man, dass diese Gleichung nicht folgt?
Ich: Man nimmt sich ein konvergentes äquivalentes TES und reduziert los, wenn die Normalformen gleich sind, dann gilt die Gleichung, sonst nicht.
Gi: Warum ist das so?
Ich: Es gilt $s \equiv_{\mathcal{E}} s \Leftrightarrow s \leftrightarrow^*_{\mathcal{E}} t \Leftrightarrow s \leftrightarrow^*_{\mathcal{R}} \Leftrightarrow s \downarrow_{\mathcal{R}} t$ mit Birkhoff, Äquivalenz und Konfluenz beziehungsweise Church-Rosser-Eigenschaft.
Gi: Ok, also richten wir die Gleichungen mal (richtet jeweils von links nach rechts). Was machen wir jetzt?
Ich: Zuerst Terminierung zeigen, dass geht mit LPO mit $app \sqsupset cons$ und der Einbettungsordnung.
Gi: Warum haben Sie LPO und nicht RPO genommen?
Ich: Weil es einfacher ist.
Gi: Ok, stimmt. Wie unterscheiden sich denn LPO und RPO?
Ich: Beim Vergleich von Termen mit gleichem äußersten Funktionssymbol. Die RPO vergleicht die Argumente als Multimenge.
Gi: Könnten wir denn die Terminierung des TES mit PRO zeigen?
Ich: Nein, da Argumente größer werden. (Gucke noch mal genau auf die Gleichung) Doch, man kann auch RPO benutzen.
GI: Können Sie beweisen, dass die Multimengenrelation fundiert ist, wenn die ursprüngliche Relation fundiert ist.
Ich: Ja (Beweis vorgeführt).
Gi: Wofür braucht man denn das Bottom?
Ich: Für die Invariante: Die Blätter des i-ten Baumes entsprechen den Elementen der i-ten Menge.
Gi: Da könnten wir aber doch die Invariante ändern. Es gibt einen anderen Grund.
Ich: Damit der Baum im jedem Schritt größer wird.
Gi: Ja, genau. Was ist, wenn man die zweite Regel andersrum orientiert, dann ist das TES ja immer noch äquivalent.
Ich: Dann müsste man Terminierung auch zeigen können, spricht eigentlich nix gegen (zeige dann mit $cons \sqsupset app$ Terminierung).
Gi: Warum haben wir die Regel dann so orientiert.
Ich: Weil wir so keine kritischen Paare haben.
Gi: Und andersrum.
Ich: Da haben wir eines, da man die erste und zweite Regel überlappen kann.
Gi: Ok, jetzt kann man also zeigen, dass die Gleichung nicht folgt.
Ich: Ja, beide Terme der Anfrage sind Normalformen, da keine Regel matcht.
Gi: Und wie kann man zeigen, dass die Gleichung für die „richtigen" Algebren

folgt?
Ich: In dem man induktive Gültigkeit überprüft (Definition aufgeschrieben).
Gi: Und wie kann man das überprüfen?
Ich: Mit der Konsistenzbeweismethode (Satz aufgeschrieben: $s \equiv t$ ist induktiv gültig, wenn durch Hinzunahme von $s \equiv t$ zu $\mathcal{E}$ keine Inkonsistenzen entstehen).
Gi: Können Sie das beweisen?
Ich: Ja, versuchen wir es mal (Beweis aufgeschrieben).
Gi: Und wie überprüft man das jetzt automatisch?
Ich: Wir haben dazu den verbesserten Vervollständigungsalgorithmus missbraucht.
Gi: Gut, kommen wir mal zum Compilerbau. Lexikalische Analyse, welche Sprachen verwendet man da?
Ich: Reguläre Sprachen.
Gi: Definieren Sie mal das erweiterte Matchingproblem.
Ich: (Definition aufgeschrieben).
Gi: Muss dass denn eindeutig sein?
Ich: Nein, für $w = aa$ und den Ausdruck a^+ gibt es mindestens zwei Zerlegungen.
Gi: Was machen wir also?
Ich: Wir benutzen das longest-match-Prinzip: Wenn man ein Match mit einem x verlängern kann, dann muss $x = \varepsilon$ sein.
Gi: Und dann ist es eindeutig?
Ich: Die Zerlegung ja, die Analyse nicht, wenn die Sprachen der regulären Ausdrücke nicht paarweise disjunkt sind. Deswegen ordnen wir die Ausdrücke und nehmen immer den mit dem kleinsten Index.
Gi: Ok, und wie sieht jetzt ein Automat aus, der uns das berechnet?
Ich: Man baut aus den n Automaten für die regulären Ausdrücke den Produktautomaten, partitioniert die Endzustände und lässt ihn dann mit zwei Köpfen laufen.
Gi: Welche Konfigurationen hat der Automat?
Ich: In der ersten Komponente T, falls schon ein Match gefunden wurde und dieses verlängert wird, oder N, falls noch kein Match gefunden wurde. Die zweite Komponente ist ein Wort aus $\Sigma^* Q \Sigma^*$ mit dem Zustand, dem Rest der Eingabe rechts und dem Teilwort seit dem letzten Match, dass wir uns für das Backtracking merken müssen. In der dritten Komponente speichern wir die bisherige Ausgabe als Sequenz von Token.
Gi: Kommen wir zur lexikalischen Analyse. Welche Sprachen benutzen wir hier?
Ich: Kontextfreie Sprachen, die sind aber zu schwach um einige Anforderungen auszudrücken.
Gi: Ja, aber bleiben wir mal bei den kontextfreien Sprachen. Ist das Wortproblem entscheidbar?
Ich: Ja, mit dem CYK-Algorithmus.
Gi: Und warum benutzt man den nicht zum Parsen?
Ich: Weil der kubische Laufzeit hat. Wir wollen aber linearen Zeitaufwand erreichen.
Gi: Und wie macht man das?
Ich: Man schränkt die Grammatiken ein: Für die Top-Down-Analyse definiert man sich die $LL(k)$-Grammatiken und für die Bottom-Up-Analyse die $LR(k)$-

Grammatiken.
Gi: Wann ist denn eine Grammatik in $LL(k)$?
Ich: (Definition aufgeschrieben) Man kann also die Entscheidung, welche Regel angewendet wird, anhand der nächsten k Symbole der Eingabe entscheiden.
Gi: Kann man entscheiden, ob eine Grammatik in $LL(1)$ ist?
Ich: Ja, dass ist der Fall, wenn die la-Mengen disjunkt sind (Definition aufgeschrieben).
Gi: Wofür braucht man hier das $fo(A)$?
Ich: Falls β zu ε abgeleitet werden kann.
Gi: Kommen wir zur semantischen Analyse. Wir haben die Regel $A \to BC$ und jedes Nichtterminal hat ein inherites und ein synthetisches Attribut. Welche Abhängigkeiten können hier auftreten.
Ich: (Skizze mit Abhängigkeiten gemalt, dabei Innen- und Außenvariablen erwähnt)
Gi: Können in dieser Situation Kreise auftreten?
Ich: Nein, da wir die Variablen partitioniert haben und es nur Pfeile von Außen- zu Innenvariablen gibt.
Gi: Und wie treten dann Abhängigkeiten auf?
Ich: Beim zusammenkleben der Regeln zu einem Ableitungsbaum.
Gi: Und wie kann man die verhindern?
Ich: Mit S-Attributgrammatiken, die nur synthetische Attribute haben und deswegen natürlich keine Kreise.
Gi: Ok, das ist dann trivial.
Ich: Dann gibt es noch die L-Attributgrammatiken. Da dürfen die inheriten Attribute nur von synthetischen abhängen, die von weiter links kommen.
Gi: Dann streichen Sie mal in der Skizze die verbotenen Abhängigkeiten.
Ich: (Einige Pfeile gestrichen, aber einen vergessen. Da ich die Definition nicht mehr genau kannte, habe ich zusammen mit Prof. Giesl dann die restlichen verbotenen Pfeile gestrichen)
Gi: Ok, dann gehen sie mal bitte nach draußen.

Die Prüfungssituation war sehr angenehm. An mehreren Stellen musste ich kurz überlegen, wie ich weiterkomme ($\leq_{log}$ transitiv, PP nicht praktisch anwendbar, Definition von L-Attributgrammatiken), was aber nach kurzen Hinweisen kein Problem war. Positiv war, dass ich den Satz zur Konsistenzmethode beweisen konnte (und man zusätzlich gemerkt hat, dass ich ihn nicht auswendig gelernt hatte).