Prüfungsprotokoll

5. April 2006

Daten:

- Prüfer: Professor Thomas
- Prüfling: Andreas Feldmann
- Themen: Angewandte Automatentheorie (2003), Model Checking (2005), Compilerbau (2004)
- Note: 1,7

AAT

Er: Wir fangen mit Automatentheorie an. Wir haben da ja die regulären Baumsprachen kennengelernt. Welche Automaten definieren diese Sprachklasse?

Ich: Die BU-NTAs, BU-DTAs und TD-NTAs.

Er: Ja, und welche Automaten fallen da raus?

Ich: Die TD-DTAs.

Er: Genau. Und können Sie auch eine Sprache nennen, die nicht von diesen Automaten erkannt werden?

Ich: Ja. *Hab dann die Sprache {f(a,b), f(b,a)} hingeschrieben.*

Er: Wie würde man denn einen Baumautomaten für reguläre Baumsprachen bauen, der die Schnitt- bzw. Vereinigungsmenge der Sprache erkennt?

Ich: Nun, man würde den Produkt- bzw. Vereinigungsautomaten bauen.

Er: Ok, und wie sieht das bei TD-DTAs aus?

Ich: *Nach einigem überlegen:* Wenn man die Sprachen {f(a,b)} und {f(b,a)} anguckt, dann kann man für jede einen TD-DTA angeben, aber vereinigen kann man sie nicht.

Er: Ok, und für die Schnittmenge kann man den Produktautomaten bauen. Wenn ich ihnen nun die folgenden drei Baumsprachen nenne, welche davon sind TD-DTA erkennbar?

schreibt auf:
1) alle Pfade haben die gleiche Länge

2) es gibt einen Pfad mir ungerader Länge
3) alle Pfade haben ungerade Länge.

Ich: Also für die erste wurde im Skript mit Hilfe des Pumpinglemmas bewiesen, dass sie nicht regulär ist. *überleg...* Für die zweite könnte man testen, ob ein Pfad gerade oder ungerade ist, indem man modulo 2 (also mit zwei Zuständen) rechnet, und dann wenn zwei Pfade zusammenkommen, man sich merkt ob einer eine ungerade Länge hatte.

die Antwort war falsch, da ich dachte, dass es inzwischen um reguläre Baumsprachen geht. Sprache 2 ist regulär, aber nicht TD-DTA erkennbar.

Er: Wie kann man denn testen ob zwei Baumautomaten äquivalent sind?

Ich: Man kann, wenn man einen NTA hat, ihn erst deterministisch machen, mit Hilfe der Potenzmengenkonstruktion, dann die DTAs minimieren und diese dann auf Isomorphie testen.

Er: Gut. Das haben wir zwar nicht so gemacht, aber wie effizient ist das verfahren denn?

Ich: Reden wir denn über NTAs oder DTAs? *Für NTAs hätte man ja schon mal einen exponentiellen Aufwand, wegen der Potenzmengenkonstruktion.*

Er: Aha. Nun, wie ist das denn bei der Minimierung für DEAs?

Ich: Das geht in polynomieller Zeit

Er: Ja, und wie bei den meisten anderen Ergebnissen für DEAs/NEAs auch, kann man dies für Baumautomaten übernehmen. Wie sieht das denn für NEAs aus?

Ich: Nun, da ist das Minimierungsproblem PSPACE-hart.

Er: Ok, und wie zeigt man das?

Ich: Das Minimierungsproblem ist ein Spezialfall des Nichtuniversalitätsproblems. *Hab dann grob die Reduktion von einem PSPACE-harten Problem auf das Nichtuniversalitätsproblem beschrieben. Wichtig war ihm, dass die Länge einer Konfiguration polynomiell eingeschränkt werden kann.*

Er: Gehen wir mal zu einem anderen Thema über: Petri Netze. Welches Problem kann man denn zB für Petri Netze entscheiden?

Ich: Das Beschränktheitsproblem.

Er: Ok, und wie ist das definiert? Und was ist wichtig bei der Definition?

Ich: Wichtig? *Ich wusste nicht worauf er hinaus wollte, und hab das dann auch gesagt.*

Er: Wichtig ist, dass man von einer Konfiguration ausgeht, denn man kann in den meisten Fällen eine Konfiguration angeben, von wo aus das Petri Netz nicht beschränkt ist.

Ich: Ach so, ja. *Hab dann das Beschränktheitsproblem definiert (ausgehend von einer Konfiguration natürlich ;).*

Er: Und wie entscheidet man das?

Ich: Man guckt ob in dem Karp-Miller Tree ein unendlich Zeichen vorkommt.

Er: Ok, und ist der Karp-Miller Tree endlich?

Ich: Ja, denn wenn ich einen unendlichen Pfad habe, dann ist das Petri-Netz dort unbeschränkt und ich kann ein unendlich Zeichen einführen. Mit den anderen Abbruchbedingungen zusammen gibt das einen endlichen Algorithmus.

Mit dieser Antwort war Herr Thomas nicht zufrieden und wollte noch wissen wie das Lemma von König und das von Dixon lauten. Diese konnte ich auch nennen, aber den Beweis für die Endlichkeit konnte ich nicht, da ich dachte, dass meine intuitive Erklärung reichen würde. Aber Herr Thomas wies mich dann darauf hin, dass wenn die intuitive Erklärung reichen würde, man die Lemmas nicht bräuchte. Wenn man sich das näher anschaut, dann ist das wohl auch gar nicht mehr so intuitiv.

Er: Kennen Sie noch ein Automatenmodell, für welches das Erreichbarkeitsproblem unentscheidbar ist?

Ich: 2-PDS

Er: Und wenn man das Modell einschränkt?

Ich: Für 1-PDS ist das Erreichbarkeitsproblem entscheidbar.

Er: Ok, und wenn man das Modell noch weiter einschränkt?

Ich: Für 1-CS ist es entscheidbar, für 2-CS nicht.

Er: Und was ist ein CS?

Ich: Ein PDS mit nur zwei Stacksymbolen, wobei man nur eines wirklich benutzt zum zählen.

Er: Und wie verhalten sich CS zu PDS?

Ich: Man kann PDS durch CS simulieren, indem man das Stackalphabet als Ziffern darstellt, und den Stackinhalt als Zahl interpretiert. Dann kann man arithmetische Funktionen benutzen um zB ein Symbol auf den Stack zu legen. *Hier hab ich wohl auch ein wenig unsauber erklärt. Er wollte hören, dass die kleinste Ziffer am Stackanfang liegt. Außerdem hatte ich vergessen zu erwähnen, dass man nicht immer zur Basis 10 rechnet:*

Er: Was passiert denn wenn man 15 Symbole hat?

Ich: Ach so ja, dann muss man zur Basis 16 rechnen.

Er: Ganz kurz noch: Wie ist denn die Idee bei der Reduktion des Halteproblems auf 2-PDS?

Ich: Man nimmt eine Konfiguration der TM und schreibt die eine Hälfte in den einen Stack und die andere rückwärts in den anderen, so dass der Lesekopf der TM immer an der Stelle der Kellerspitzen liegt.

MC

Er: Gehen wir mal zum Model Checking über. Können Sie mir sagen wie das synchronisierte Produkt zweier Transitionssysteme aussieht?

Ich: *Definition der Transitionsrelation des synchronisierten Produkts erklärt.*

Er: Ok, sie haben ja Safety und Liveness Properties kennengelernt. Wie sind die denn definiert?

Ich: *formale Definition von Safety Property angegeben.*

Er: Kann man das auch irgendwie einfacher ausdrücken?

Ich: Nun, es darf keine Bad Prefixes geben.

Er: Genau, so merke ich mir die Definition; es darf nur gute Präfixe geben. Diese formale Formulierung kann man sich doch nicht merken. Wie ist das denn mit Liveness Properties?

Ich: *formale Definitionen von Liveness Property angegeben; einmal mit Präfixmenge und einmal mit Abschluss.*

Er: Und wie kann man das intuitiver formulieren?

Ich: Für jedes endliche Wort gibt es ein unendliches, das hinten angehängt werden kann, so dass das resultierende Wort in P_life liegt.

Er: Ok, gibt es Properties, die sowohl Liveness als auch Safety sind?

Ich: Ja, eine. Nämlich die Menge aller unendlichen Wörter. *Hab noch erklärt warum (siehe MC Vorlesung von Prof. Katoen).*

Er: Wie würden Sie denn eine Safety Property prüfen?

Ich: Also erstmal sollte die Safety Property regulär sein... *Wollte das Verfahren mit NEAs für die MinBadPrefix Menge erklären.*

Er: Moment, das muss doch nicht regulär sein...

Ich: Öhh... *gedanklich: nicht???? dann weiß ich aber auch nicht wie das gehen soll...*

Er: Kennen Sie Invarianten?

Ich: Ja. *Definition einer Invarianten hingeschrieben.*

Er: Ja, und wie kann ich damit Safety Properties prüfen?

Ich: *Wusste nicht so recht was er von mir wollte, daher beschreibe ich das Verfahren für reguläre Safety Properties und erkläre, dass man das auf invarianten Prüfung zurückführt (bei der Tiefensuche im TS).*

Er: *nickt.* Man kann auch Induktiv vorgehen und jede Safety Property mit Hilfe von Invarianten prüfen, indem man immer einen Schritt des TS macht, und guckt ob noch alles in Ordnung ist.

Ich: Aha, Ok. *gedankliches Achselzucken...*

Er: Sie haben ja auch das Model Checking mit LTL kennengelernt. Wie sieht denn der Zustandsraum des NBAs aus, den man aus einer LTL Formel herleitet?

Ich: Das sind diese elementary Sets, die Teilmengen der Abschlussmenge der Formel sind. *Hab dann angefangen die Eigenschaften der elementary Sets aufzulisten, aber Herr Thomas hat mich dann unterbrochen:*

Er: Und was für eine Komplexität hat die Umwandlung der Formel in einen NBA?

Ich: *Hier hab ich sehr ungenau geantwortet, so dass es so klang, als würde das in Polynomzeit gehen, obwohl mir klar war, dass dies der Schritt ist, der das ganze PSPACE hart macht. Nachdem Herr Thomas mich dazu gebracht hatte zu erklären, dass nach der Umwandlung nur noch eine Tiefensuche ansteht, die ja bekanntlich in linearer Zeit machbar ist, hat er mir erklärt, dass das Umwandlungsproblem folglich PSPACE-hart sein muss. (siehe auch Anmerkungen unten.)*

CB

Er: Gehen wir zum Compilerbau über. In der Syntaxanalyse gibt es verschiedene Grammatiken, die wir verwenden, welche sind das?

Ich: LL(k) und LR(k). Diese benutzt man, um ein Verfahren zu kriegen, das in linearer Zeit lösbar ist. Mit LL(k) Grammatiken berechnet man eine TD-Analyse, und mit LR(k) Grammatiken eine BU-Analyse.

Er: Wie kann man denn testen, ob eine Grammatik LL(k) ist?

Ich: Man guckt ob, die Schnittmenge der la-Mengen der Regelalternativen leer ist.

Er: *War damit nicht ganz zufrieden.* Wie ist denn die la-Menge definiert?

Ich: *Definition der la- und fi-Mengen hingeschrieben. Bei der first Menge steht ja so etwas wie $\alpha \Rightarrow^* vw$ in der Definition. Daher:*

Er: In der first Menge müssten Sie ja jetzt alle Ableitungen von α betrachten. Aber wie können Sie das denn Algorithmisch testen?

Ich: *Erkläre etwas unpräzise das Verfahren wie man die first Mengen aus einer Grammatik berechnet, so wie das im Skript auch gemacht wird.*

Er: Ok, das werte ich jetzt mal als Algorithmus. Was fallen Ihnen für Dinge ein, die man aus einer LR(k) Grammatik herausziehen kann, um die Analyse deterministisch zu machen?

Ich: LR(k)-Auskünfte.

Er: Ja, oder LR(k)-Mengen. Wie ist denn eine LR(k)-Auskunft definiert?

Ich: LR(0) oder LR(1)?

Er: Was passiert denn bei LR(2) zB?

Ich: Nun, eine LR(2) Grammatik kann keine größere Sprachklasse als die LR(1) Grammatik definieren, daher bringt einem das nichts.

Er: Ja, das haben Sie aber wahrscheinlich nicht bewiesen, oder?

Ich: Nein, das habe ich nur in einem der Protokolle gelesen, dass das wohl so ist.

Er: *lacht* Also schreiben Sie mal die Definition einer LR(0) Grammatik auf.

Ich: *schreibe auf...*

Er: Und wie arbeitet denn der BU-Automat?

Ich: Also die LR(0)-Informationen sind das Stack Alphabet, und man benutzt die action Funktion um zu wissen, ob man shiften oder reduzieren soll...

Er: Ok, und es gibt noch eine Funktion, welche ist das?

Ich: Die goto Funktion, die einem sagt, welche LR(0)-Information man auf den Keller schreiben muss, wenn man shiftet...

Er: Kennen Sie ein Verfahren mit dem man die LR(0) Informationen berechnen kann?

Ich: Man kann einen Automaten bauen, dessen Zustände aus den LR(0) Auskünften bestehen. Die Transitionen beschreiben, wie man von einer Auskunft zur nächsten kommt. Dann baut man daraus den Potenzmengenautomaten und die Zustände sind dann die LR(0) Mengen, und die Transitionsfunktion ist die goto Funktion.

Er: Gut, können Sie dann mal draußen warten?

Anmerkungen

Als Vorbereitung habe ich das AAT Skript von 2003, die CB Mitschrift von 2004 und die MC Lecturenotes (also das Buch von Prof. Katoen) benutzt. Die AAT und MC Vorlesungen habe ich besucht, für CB hab ich mir die Videos angeguckt. Ich habe alle Übungen zu den Vorlesungen gerechnet, und danach erstmal alles Wissenswerte aus den Unterlagen rausgeschrieben und auswendig gelernt. Danach bin ich die Prüfungsprotokolle durchgegangen und habe versucht alle Fragen zu meinen Themen zu beantworten (hab aber zeitlich nicht alle geschafft). Das ganze hat mich vier Monate Zeit gekostet. Ich habe auch ein wenig mit anderen Leuten gelernt, aber nicht allzuviel, da ich nicht viele gefunden hab, die mit mir lernen wollten.

Herr Thomas hat mir hinterher erklärt was mir für die 1,0 gefehlt hat. Ich habe wohl drei grosse Patzer gemacht. Das erste war, dass ich nicht erkannt hatte, dass die Baumsprache der Bäume, die einen Pfad ungerader Länge besitzen, nicht TD-DTA erkennbar ist. Das zweite war, dass ich den Beweis für die Endlichkeit des Karp-Miller Trees nicht konnte. Und das Dritte war, dass ich nicht wusste, dass die Umwandlung einer LTL Formel in einen NBA PSPACE-hart ist. Herr Thomas meinte auch, dass ein solcher Fehler akzeptabel ist, aber drei waren einfach zu viel. Dazu muss ich sagen, dass der erste Patzer auf das Missverständnis zurück ging, dass ich dachte es ginge um reguläre Baumsprachen. Das hat sich auch erst im nachhinein geklärt, als mich jemand darauf hinwies, dass diese Baumsprache doch regulär ist. Beim zweiten Fehler, wusste ich den Beweis nicht, da ich dachte, dass das eh intuitiv zu erklären ist. Ich hatte mir den Beweis irgendwann mal angeguckt und ihn aber für ziemlich kompliziert gehalten. Danach hab ich nie mehr draufgeguckt, und dachte, dass das wohl eh recht unwahrscheinlich ist, dass er danach fragt. Ich hab wohl ein wenig gepokert, und das war mir auch bewusst. Von daher muss ich damit wohl leben. Beim dritten Patzer habe ich das wohl schlecht erklärt, so dass es so klang als ob ich das

nicht wüsste. Ich hätte wohl mehr darauf bestehen sollen, dass mir das Problem bewusst ist. Das war wirklich doof, und dafür muss ich mir wohl in den Arsch beißen. Fazit ist: wenn man eine besonders gute Note haben will, sollte man alles ausnahmslos lernen. Auch Dinge die einen nicht motivieren (es reicht in der Prüfung ja meistens schon die Idee, die dahintersteckt, erklären zu können). Außerdem sollte man die Dingen auch gut erklären können um keine Missverständnisse zuzulassen. Falls das doch passiert, besser nochmal nachhaken und es gerade biegen.

Auch wenn das hier im Protokoll vielleicht so aussieht, als ob ich die meiste Zeit nur geredet hätte, habe ich immer auch gleichzeitig was dazu aufgeschrieben oder gemalt. Das hilft mir meine Gedanken zu ordnen. Zum Schluss waren drei Seiten voll gekritzelt. Und natürlich habe ich auch immer was erzählt, wenn im Protokoll nur steht, ich hätte was hingeschrieben.

Wenn Du selbstsicher rüberkommen willst, kann ich dir einen Tip geben, den meine Stiefmutter mir am Tag vor der Prüfung noch gegeben hat: Guck dem Prüfer auf die Nase. Auf keinen Fall immer nur nach unten gucken und vor sich her murmeln. Das kommt nicht gut an.

Dann wünsche ich noch viel Glück, und schreibt Protokolle Leute...