Diplomprüfung Vertiefungsfach Automaten, Logik und Verifikation

9. Mai 2006

Prüfungsthemen waren

- Automaten auf unendlichen Wörtern (Prof. Thomas, 2 SWS, WS 04/05)
- Baumautomaten und Anwendungen (Dr. Löding, 2 SWS, WS 04/05)
- Advanced Theory Of Finite Automata (Prof. Thomas, 2 SWS, SS 05)
- Unendliche Transitionssysteme (Dr. Löding, 2 SWS, SS 05)
- Model Checking (Prof. Katoen, 4 SWS, WS 05/06)
- Symbolisches CTL Model Checking aus der Vorlesung Model Checking (Prof. Thomas, WS 03/04)

Prüfer waren Prof. Thomas, der mich über Automaten auf unendlichen Wörtern, Advanced Theory Of Finite Automata und Model Checking geprüft hat, sowie Dr. Löding, der entsprechend für Baumautomaten und Anwendungen und Unendliche Transitionssysteme zuständig war.
Vorbereitet habe ich mich etwas mehr als zwei Monate, hauptsächlich mit meinen Mitschriften aus den Vorlesungen bzw. mit dem Folienskript von Prof. Katoen. Außerdem waren die Videos der Vorlesungen Automaten und Reaktive Systeme und Angewandte Automatentheorie von der Internetseite des Lehrstuhl sehr nützlich.
Vor und zu Beginn der Prüfung war ich sehr nervös. Nach den ersten Minuten war die Nervosität dann aber verschwunden. Sowohl Prof. Thomas als auch Dr. Löding empfand ich als sehr angenehme Prüfer.
Die Reihenfolge des Protokolls entspricht der der Prüfung (auch wenn sie ein bisschen durcheinander ist).
Die Prüfung dauerte etwas länger als 50 Minuten. Ich hoffe alles richtig wiedergegeben zu haben, kann aber natürlich nichts garantieren.

Angewandte Automatentheorie

Fangen wir mit der Minimierung von DEAs und NFAs an. Was wissen Sie dazu?
Zunächst DEA Minimierung erklärt: Zustandsäquivalenz, Block-Splitting Algorithmus, Komplexität $O(|Q|^2 \cdot |\Sigma|)$ (auf Nachfrage $O(|Q| \cdot \log |Q|)$).
Problematik bei NEA-Minimierung angesprochen, PSPACE-Vollständigkeit inkl. Beweisidee, Bisimulation, Algorithmus, Komplexität.

Wie würde man den den minimalen NEA konstruieren? Gibt es eigentlich den minimalen NEA?
Es gibt Beispiele für Sprachen, für die unterschiedliche minimale NEAs existieren. *Aber wie konstruiert man den?* Ich habe dann die Idee zu dem universellen NEA aus dem Skript über Angewandte Automatentheorie erzählt (zumindest das, an das ich nach einmaligem Überfliegen des Abschnitts noch erinnern konnte).

Wir haben dann Baumautomaten über endlichen Bäumen kennengelernt. Wie funktionieren die denn?
Funktionsweise der Baumautomaten erklärt, insbesondere die Transitionsfunktion / -relation und Akzeptanzbedingungen.

Ist die folgende Baumsprache regulär und wie würden Sie ggf. einen Baumautomaten konstruieren? „Es gibt genau einen Pfad, der nur aus c's besteht."
Ja, die Sprache ist regulär. Ich würde einen Top-Down Baumautomaten konstruieren, der den Pfad rät und überprüft, ob der Pfad nur aus c's besteht und ob es auch wirklich der einzige ist.

Es gibt ja auch eine Logik über diesen Bäumen, was können Sie denn dazu sagen?
Das kam so in der Vorlesung nicht dran... Die MSO-Logik? Vom Wortfall und der Logik S2S auf den Fall über endlichen Bäumen geschlossen. Zusätzlich wollte Prof. Thomas die atomaren Formel wissen.

Wie würde man denn in der Logik die Baumsprache von vorhin formulieren?
Man müsste wohl die Existenz eines Pfades fordern und sagen, dass dieser der einzige Pfad ist, der ausschließlich aus c's besteht.

Ja, das geht wohl. Aber benötigt man wirklich Quantoren zweiter Stufe?
Ich habe noch eine Idee geäußert, wie man das ausschließlich mit Variablen erster Stufe ausdrücken könnte. Prof. Thomas hat mir dann erklärt, dass man dazu von den Blättern anfängt und mit „Vorgängerabschluss" arbeitet.

Gut, dann wird Herr Löding Ihnen jetzt ein paar Fragen zu den unendlichen Transitionssystemen stellen.

Unendliche Transitionssysteme

Wir haben als erstes die rekursiven Automaten behandelt, wie sehen die denn aus?
Die Definition für einen rekursiven Automaten ausgeschrieben, insbesondere die Ein- und Ausgangszustände, sowie die Transitionsrelation erklärt.

Ist das Punkt-zu-Punkt Erreichbarkeitsproblem entscheidbar?
Ja, dazu zunächst die naive und dann die „clevere" Idee der Reduktion zu Pushdown-Systemen erklärt. Das Problem für PDS ist dann entscheidbar.

Und warum macht man sich damit so Mühe?
In die Komplexität der meisten Algorithmen geht die Anzahl der Zustände stärker ein als die Anzahl der Stacksymbole. Die Komplexität des Saturierungsalgorithmus ist $O(|P| \cdot |\Gamma| \cdot |\Delta| \cdot |Q|^3)$.

Kommen wir zu den Petrinetzen. Welche Probleme haben wir denn da in der Vorlesung besprochen?
Beschränktheit, Deadlock-Freiheit und das Überdeckungsproblem beschrieben.

Wie löst man denn das Überdeckungsproblem?
Mit dem Karp-Miller-Baum. Kurz die Idee des Karp-Miller-Baums beschrieben und wie man ihn berechnet, insbesondere die Verwendung des ∞-Zeichens. Eine Markierung m ist genau dann in einem Petrinetz von einer Anfangsmarkierung m_{in} aus überdeckbar, wenn es eine erweiterte Markierung m_+ mit $m_{in} \rhd^* m_+$ in dem Karp-Miller-Baum gibt, für die $m \leq m_+$ gilt.

Gut. Dann gabs ja noch die Registermaschinen / Zählersysteme. Ist denn da das Überdeckungsproblem entscheidbar?
Gute Frage... Also, für 1-Registermaschinen ist das Problem durch Reduktion auf PDS entscheidbar. Für n-Registermaschinen denke ich, dass es nicht entscheidbar ist. Vielleicht kann man irgendwie das Erreichbarkeitsproblem darauf reduzieren ... ?!?

Na gut, Ihre Intuition ist richtig. Das soll mal reichen.

Automaten auf unendlichen Wörtern

Von den unendlichen Transitionssystemen zu den Automaten auf unendlichen Wörtern. Es gibt da ja mehrere Akzeptanzbedingungen. Welche Sprachen werden denn von den Automaten mit der Akzeptanzbedingung „es gibt einen unendlichen Lauf" erkannt?
Dazu konnte ich leider nicht so viel sagen. Mit Hilfe kam ich darauf, dass es etwas mit E- bzw. A-Akzeptanz zu tun hat. Ich habe dann noch etwas zu Garantie- und Sicherheitseigenschaften erzählt.

Wie kann man denn die Sprachen charakterisieren, die sowohl E- als auch A-erkennbar sind? Kennen Sie eine solche Sprache?
Die Sprache $1\mathbb{B}^\omega$ ist z.B. eine solche Sprache. *Aber wie lassen sich E- und A-erkennbare Sprachen charakterisieren?* Nach ein wenig Nachdenken: Natürlich! Das sind alle Sprachen, die einen festen endlichen Anfang haben.

Kann man algorithmisch entscheiden, wann Sprachen E- bzw. A-erkennbar sind wenn man einen ω-regulären Ausdruck gegeben hat?
Aus dem ω-regulären Ausdruck konstruiert man einen nichtdeterministischen Büchi-Automaten, dann per Muller-Schupp einen deterministischen Rabin-Automaten und daraus einen deterministischen Muller-Automaten. Nach Satz von Landweber kann man dann anhand der Schleifenstruktur der Akzeptanzkomponente entscheiden, ob die Sprache E- oder A-erkennbar ist.

Und wie geht das dann genau?
Bei E-Erkennbarkeit ist die Akzeptanzkomponente unter erreichbarer Schleifen abgeschlossen. *Aber wie ist das bei A-erkennbaren? Vielleicht unter „erreichten Schleifen"?*

Man muss das für A-erkennbare Sprachen nicht extra definieren. Es gibt da einen Satz!
Richtig, eine Sprache L ist E-erkennbar genau dann, wenn $\Sigma^\omega \setminus L$ A-erkennbar ist. Dann braucht man den entsprechenden Teil des Satzes von Landweber wirklich nicht.

Betrachten wir mal die folgenden Automatenmodelle: nichtdeterministische Büchi-Automaten, deterministische Muller-Automaten und nichtdeterministische Muller-Automaten. Wie sieht es da jeweils mit der Ausdrucksstärke aus?
Die sind alle drei gleich ausdrucksstark. Von nichtdeterministischen Büchi-Automaten kommt man mit der Muller-Schupp-Konstruktion zu deterministischen Rabin-Automaten und von da aus zu den deterministischen Muller-Automaten. Aus einem deterministischen Muller-Automaten konstruiert man einen nichtdeterministischen Büchi-Automaten, indem man zuerst rät wann nur noch Zustände aus $inf(\rho)$ gesehen werden und dann mit welcher Akzeptanzmenge der Muller-Automat akzeptieren würde. Diese sammelt man dann immer wieder Zustände auf.

Und wie funktioniert die Konstruktion mit den nichtdeterministischen Muller-Automaten?
Das geht genauso wie bei deterministischen Muller-Automaten. Da der Büchi-Automaten sowieso nichtdeterministisch ist, macht das keinen Unterschied.

Wie ist die Komplexität der Muller-Schupp-Konstruktion?
Der aus dem nichtdeterministischen Büchi-Automat mit $O(n)$ Zuständen resultierende deterministische Rabin-Automat hat $2^{O(n \cdot \log n)}$ Zustände und $O(n)$ Akzeptanzpaare.

Warum ist das denn so „schwer“?
Das Problem ist, dass man Büchi-Automaten nicht einfach so Determinisieren kann. Die deterministischen Büchi-Automaten sind nicht so ausdrucksstark wie die nichtdeterministischen.

Model Checking

Betrachten wir das LTL Model Checking. Wie konstruiert man denn aus einer LTL-Formel einen Automaten?
Ich habe die Idee der Konstruktion in einen verallgemeinerten Büchi-Automaten erklärt. Recht detailliert die elementary sets, die Transitionsrelation und habe die Akzeptanzbedingung hergeleitet. Auch wenn das hier recht kurz aussieht, hat es eine gute Weile gedauert.

Und wie groß wird der resultierende Büchi-Automat?
Im schlimmsten Fall ist bekommt man als Zustände alle Teilmengen von $cl(\varphi)$. Das ergibt dann $O(|\varphi| \cdot 2^{2|\varphi|})$.

Wie ausdrucksstark ist denn LTL?
Durch die Konstruktion eines Büchi-Automaten haben wir ja schon gezeigt, dass jede LTL-definierbare Sprache auch ω-regulär ist. Aber nicht jede ω-reguläre Sprache ist auch LTL-definierbar, z.B. die Sprache $\{\sigma \in (2^{AP})^\omega \mid a \in \sigma[2i], 0 \leq i\}$. *Es gab dann eine kleine Diskussion darüber, ob die Sprache nicht doch LTL-definierbar ist. Herr Löding und ich konnten Prof. Thomas aber davon überzeugen, dass diese Sprache wirklich nicht LTL-definierbar ist, die Sprache, in der in genau jedem zweiten Buchstaben ein a vorkommt, allerdings schon.*

Und was können sie zur Komplexität des LTL Model Checking sagen?
Das LTL Model Checking ist PSPACE-vollständig.

Können Sie das beweisen?
Nein, das kann ich leider nicht. Weder Sie noch Herr Katoen haben das in der Vorlesung gemacht. Ich kann Ihnen alternativ aber beweisen, dass das Problem NP-schwer ist. *Eigentlich wollte ich noch schnell nachschieben, dass ich ihm aber wenigstens die Idee des PSPACE-Vollständigkeitsbeweis liefern könnte. Aber bevor ich das erwähnen konnte, ging es schon weiter...*

Dann zeigen Sie doch, dass das LTL Model Checking NP-schwer ist!
Möchten Sie Ihre Reduktion von 3-SAT auf das LTL Model Checking oder die Reduktion von Herrn Katoen von Hamiltonkreis?

Das ist mir egal.
Ich habe dann die Reduktion von Hamiltonkreis auf das LTL Model Checking erklärt. Auf die Konstruktion des Transitionssystems aus dem gegebenen Graphen bin ich dabei nur recht grob eingegangen, habe aber die Formel $\varphi = \neg\left[\bigwedge_{v \in V}\left(\Diamond v \ \wedge \ \Box(v \rightarrow O\Box\neg v)\right)\right]$ aufgeschrieben und erklärt.

Kommen wir zum symbolischen CTL Model Checking. Was ist denn die Idee dabei?
Die Idee ist, nicht auf der expliziten Darstellung der Mengen zu arbeiten, sondern auf Ebene der Mengenbeschreibungen. Dazu stellt man das Transitionssystem bzw. die Kripke-Struktur durch boolesche Funktionen dar. Aus diesen wird dann induktiv über den Formelaufbau die charakteristische Funktion der LTL Formel berechnet. *Auf die Berechnung im einzelnen musste ich aber nicht eingehen.*

Sie sprachen gerade von einer Darstellung des Transitionssystems als boolesche Funktionen. Wie genau werden diese denn dargestellt?
Man verwendet OBDDs. Habe dann näher erklärt, was OBDDs sind und grob beschrieben, was man für Operationen auf den OBDDs für das Model Checking ausführen muss.

Und wie ist hier die Komplexität?
Das symbolische CTL Modecl Checking ist PSPACE-vollständig. *Es reichte aber nur zu sagen, dass das Problem PSPACE-vollständig ist.*

Bei Herrn Katoen haben Sie ja die Linearzeit-Eigenschaften kennengelernt. Erklären Sie doch mal, was Safety- und was Liveness-Properties sind.
Habe die Definition von Safety- und Liveness-Properties aufgeschrieben und umgangssprachlich erklärt.

Was kann man denn über den Zusammenhang zwischen diesen beiden Arten der Linearzeit-Eigenschaften sagen?
Jede Linearzeit-Eigenschaft P lässt sich als Schnitt einer Safety- und einer Liveness-Property darstellen, nämlich $P = cl(P) \cap \big(P \cup (2^{AP})^\omega \setminus cl(P)\big)$.

Ist der Beweis schwer?
Nein, soll ich es beweisen?

Nein, kommen wir noch kurz auf das „normale“ CTL Model Checking zu sprechen. Wie funktioniert das denn?
Die Idee ist die gleiche wie beim symbolischen CTL Model Checking, nur diesmal arbeitet man auf den expliziten Mengenbeschreibungen. Man berechnet ebenfalls induktiv über den Formelaufbau die Mengen $Sat(\Psi)$ für alle Teilformeln Ψ von Φ.

Letzte Frage: Welche Komplexität hat das CTL Model Checking?
Alle Algorithmen laufen linear in der Größe des Transitionssystems. Insgesamt ergibt sich eine Komplexität von $O(|TS| \cdot |\Phi|)$.

Baumautomaten und Anwendungen

Wir haben unterschiedliche Automatenmodelle kennengelernt, z.B. Muller- und Paritäts-Baumautomaten. Was können Sie denn zu deren Ausdrucksstärke sagen?
Muller- und Paritäts-Baumautomaten haben die gleiche Ausdrucksstärke. Die Richtung von Paritäts-Baumautomaten zu Muller-Baumautomaten ist einfach, man verwendet den gleichen Automaten mit der Akzeptanzkomponente $\mathcal{F} = \{P \subseteq Q \mid \exists i : F_i \cap P \neq \emptyset \wedge E_i \cap P = \emptyset\}$.

Und wie funktioniert die Richtung von Muller zu Parität?
Da verwendet man die LAR-Konstruktion. Habe dann die Idee und die LAR_Q-Mengen beschrieben. Man bekommt $n!$ viele Zustände, es reicht also nicht sich nur eine Menge von Zuständen zu merken, man benötigt mehr Informationen (etwa noch die Reihenfolge).

Paritäts-Baumautomaten sind ja unter Komplement abgeschlossen. Wie zeigt man das denn?
Man verwendet dazu Paritätsspiele. Aus einem Baumautomaten $\mathfrak{A}$ und einem Eingabebaum t konstruiert man das Paritätsspiel $\mathcal{G}_{\mathfrak{A},t}$. Der Automat akzeptiert den Baum genau dann, wenn der Spieler Automat in dem Spiel eine positionale Gewinnstrategie hat. Jetzt nutzt man die Determiniertheit von Paritätsspielen. Eine entsprechende Gewinnstrategie für den Spieler Pfadfinder kann man raten und verifizieren, dass es auch wirklich eine ist. *Mehr musste ich dazu aber nicht erzählen.*

Angenommen man hat einen Paritäts-Baumautomat mit $O(n)$ Zuständen und $O(k)$ Farben. Kann man da was über die Anzahl der Farben sagen, die der Komplementautomat haben muss?
Sehr gute Frage... damit konnte ich nicht wirklich viel anfangen. Das hat Herr Löding auch gemerkt und mir weitergeholfen.

Wenn man einen Büchi-Baumautomaten hat, wie viele Farben hat dann der entsprechende Paritäts-Baumautomat?
Man kommt mit den Farben 1 und 2 aus.

Richtig. Und was wissen Sie über das Komplement von Büchi-Baumautomaten?
Die sind nicht unter Komplement abgeschlossen. Habe dann noch die Baumsprache erwähnt, mit der man das zeigen kann. *Irgendwie half mir das aber auch nicht viel weiter.*

Wenn man jetzt einen Paritäts-Baumautomaten mit nur zwei Farben hat, kann man die Baumsprache dann vielleicht schon mit einem Büchi-Baumautomaten erkennen?
Ja, das geht. *Und dann dämmerte es mir.* Nimmt man die Baumsprache von oben (mit der man zeigt, dass die Büchi-Baumautomaten nicht unter Komplement abgeschlossen sind), dann kann muss der entsprechende Paritäts-Baumautomat für die Komplementsprache mehr als nur zwei Farben haben.

Richtig! Im Allgemeinen kann man die Anzahl der Farben nur durch $O(n \cdot k)$ beschränken.
Wie ist denn die Komplexität der Komplementierung?
Ich habe irgendwas von doppelt exponentiell erzählt. Herr Löding meinte dann, dass man es auch einfach exponentiell schaffen könnte, wenn man anstatt eines Muller-Automats wie in der Vorlesung einen Büchi-Automaten verwenden würde.

Als letztes noch kurz zur Synthese. Was können Sie denn dazu sagen?
Habe dann das Syntheseproblem erklärt. Ich habe eigentlich alles bis zur Konstruktion der Automaten erzählt. Insbesondere auch wie eine solche Funktion definiert ist, was die Probleme sind und dass man auf Bäume zurückgreift.

Warum wählt man nicht einfach eine Funktion f mit $L(f) = \emptyset$?
Solche Funktionen gibt es nicht. Eine Funktion muss zu einer Eingabe auch immer eine Ausgabe produzieren.

OK, letzte Frage. Wie siehts mit der Komplexität aus?
Der Automat hat $2^{2^{O(|\varphi|)}}$ Zustände und $2^{O(|\varphi|)}$ Farben.