Gedächtnisprotokoll zur mündlichen Diplomprüfung

im Vertiefungsgebiet Theoretische Informatik

Automaten, Logik und Verifikation

12. August 2010 (SS 2010, RWTH Aachen)

Florian Weingarten ⟨flo@hackvalue.de⟩

Inhalt:

- Angewandte Automatentheorie (V4, nach Skript von 2005)
- Automaten und reaktive Systeme (V4, nach Skript von 2003)
- Introduction to Model Checking (V4, nach Buch von Prof. Katoen bis einschließlich Kapitel 7.6)

Prüfer: Prof. Dr. Wolfgang Thomas (Informatik 7: Logik und Theorie diskreter Systeme)

Prüfungsdauer: ca. 50 Minuten

Prüfungsnote: 1.0

Achtung: Hierbei handelt es sich *nicht* um ein offizielles Prüfungsprotokoll der RWTH sondern um ein privates Gedächtnisprotokoll, das ca. fünf Stunden nach der Prüfung angefertigt wurde. Ich habe mit Sicherheit einige Sachen vergessen und gebe natürlich keine Garantie auf Korrektheit.

Inhalt: Ich habe im Sommersemester 2008 die Vorlesung *Angewandte Automatentheorie*[1] von Prof. Thomas und im Wintersemester 2008/2009 die Vorlesungen *Unendliche Spiele*[2] von Prof. Thomas sowie *Automaten auf unendlichen Wörtern*[3] von Dr. Löding gehört (die beiden letzten Vorlesungen entsprechen inhaltlich genau der ehemaligen Diplom Vorlesung *Automaten und reaktive Systeme*). Ich habe mich vor der Prüfung mit Prof. Thomas darauf geeinigt, dass ich nach dem Inhalt aus den Skripten geprüft werde und nicht nach den Vorlesungen (Folien), da in den Folien ein paar Sachen vorkamen, die ich nicht so schön fand und im Skript dafür andere Sachen, die mir besser gefallen haben. Herr Thomas war da sehr aufgeschlossen und unproblematisch.

Angewandte Automatentheorie

WT: Kennen Sie gewichtete Automaten?

Hier war ich direkt etwas geschockt. Mir ist dann eingefallen, dass diese im Skript gar nicht vorkommen und ich habe Prof. Thomas darauf hingewiesen :-)

WT: Achso, ja, dann fangen wir halt mit Bäumen an. Sagen Sie mir mal drei verschiedene Arten wie man reguläre Baumsprachen charakterisieren kann.

FW: Erstmal kann man sagen, dass eine Baumsprache regulär ist genau dann, wenn es einen regulären Baum-Ausdruck gibt, der sie erzeugt. Dann hatten wir Baum-Automaten. Da hatten wir ein paar Variationen und da muss man etwas aufpassen welche man nimmt, denn die deterministischen Top-Down Varianten sind nicht äquivalent zu regulären Ausdrücken. Und als dritte Möglichkeit könnte man eine Logik nehmen.

WT: Eine Logik? Welche?

FW: Analog zu dem Fall von Wörtern nimmt man die MSO Logik.

WT: Wie macht man denn aus einem Baum-Automaten so eine Logik Formel?

FW: Sagen wir mal ihr Automat ist ein nicht-deterministischer Bottom-Up Baum-Automat. Das geht dann ebenfalls genauso wie bei Wörtern. Man drückt aus, dass es einen akzeptierenden Lauf gibt, denn das ist ja genau die Akzeptanzbedingung bei Bottom-Up-NTA.

WT: Wie genau?

FW: Naja, man braucht dann für jeden Zustand im Automaten einen Existenzquantor und eine Mengenvariable. Dann sagt man, dass die Mengen das Universum partitionieren und so weiter. Alles wie bei Wörtern. Man muss dann noch gucken, dass die Übergänge alle mit der Übergangsrelation kompatibel sind.

[1] heißt im Bachelorstudiengang mittlerweile *Applied Automata Theory* und enthält auch den Stoff der Vorlesung *Tree Automata* bzw. *Baumautomaten*
[2] aka. *Infinite Games*
[3] aka. *Infinite Computations* oder *Automata on Infinite Words*

WT: Jaja, genau. Ist das bei dieser Konstruktion egal, ob man einen NTA oder DTA nimmt?

FW: Hm. Wir haben das für NTA gemacht. Aber man kann natürlich jeden DTA als NTA auffassen, der keinen Gebrauch vom Nichtdeterminismus macht. Die Formel die dabei rauskommt, wäre natürlich trotzdem korrekt.

WT: Ja genau!

Ich hatte das Gefühl, dass er hier darauf hinauswollte, dass man jeden NTA in einen DTA umwandeln kann, obwohl das natürlich eigentlich trivial ist.

WT: Okay, jetzt hat man hier lauter Existenzquantoren. Was meinen Sie... Geht das auch ohne?

FW: Hm. Wie, ohne Set-Quantoren?

WT: Nein nein, aber nur mit Allquantoren.

Mir war zuerst nicht klar wie das gehen sollte und ich habe dann auch bestimmt eine Minute lang überlegt und niemand hat ein Wort gesagt. Fand es sehr angenehm, dass er mich in Ruhe hat überlegen lassen.

FW: Ach so, ja, klar. Man kann einfach einen DTA nehmen. Dann sagt man sowas wie „**Jeder** korrekte Lauf ist akzeptierend“. Da es bei DTA ja immer nur einen Lauf gibt auf einem festen Baum, ist das in Ordnung.

WT: Ja, sehr schön!

Diese Idee kam, soweit ich weiss, weder in der Vorlesung noch im Skript noch in den Übungen vor und Herr Thomas hat sich sehr gefreut, dass ich darauf gekommen bin :-)

WT: Okay, Sie hatten noch was von regulären Ausdrücken gesagt. Wie ist die Konkatenation definiert?

FW: Man braucht bei der ersten Sprache an den Blättern solche Markierungspunkte. Da hängt man dann die Bäume aus der zweiten Sprache rein, also, der Markierungspunkt im ersten wird durch die Wurzel vom zweiten Baum ersetzt.

WT: Gibt es da immer nur eine Markierung? Oder kann es mehrere geben?

FW: Nein es darf ruhig mehrere geben. Man hängt dann an jede Markierung einen rein. Am Ende muss aber jeder Punkt ersetzt sein, es dürfen keine mehr übrig sein.

WT: Ja genau. Gut, betrachten wir mal ein paar Entscheidungsprobleme. Können Sie algorithmisch testen ob eine Baumsprache unendlich ist?

FW: Hm. *Habe wieder eine Minute lang überlegt und er hat mich auch nachdenken lassen und nicht gedrängt.* Also ich kann ja Leerheit testen und man hat alle wichtigen Abschlusseigenschaften. Ich vermute, dass das jetzt irgendwie einfacher ist, wenn man das Komplement anguckt.

WT: Nein nein, bleiben Sie mal bei der Sprache. Ich gebe ihnen mal einen Tipp: Wenn die Sprache unendlich ist, dann gibt es Bäume beliebiger Höhe.

FW: Das klingt sehr als wollten Sie auf das Pumping Lemma hinaus. Das sagt ja, dass wenn man einen Baum nimmt, der höher als n ist, dass man dann immer eine Zustandswiederholung hat an der man den mittleren Teil aufblasen kann.

WT: Ja genau. Und weiter?

Bin nicht auf die Lösung gekommen, aber Herr Thomas hat es dann erklärt. Man betrachtet die Bäume der Höhe $n \leq h \leq 2n$ und kann dann irgendwie mit dem Pumping Lemma argumentieren. Ich war etwas nervös und hab es nicht ganz verstanden, aber war dann auch egal.

WT: Diese Bäume sind ja immer von beschränkter Verzweigung. Wie kann man das umgehen?

FW: Mit XML-Automaten. Da hat man unbeschränkte Verzweigung. Formal sieht das einfach so aus, dass in der Transitionsrelation ein regulärer Ausdruck über Q steht und die Kinderknoten dann darauf matchen müssen.

WT: Gut. Nehmen wir mal die Sprache der boolschen Ausdrücke mit $\wedge$ und $\vee$, die zu 1 evaluieren. Die Operatoren sind ja binär, aber wegen der Assoziativität kann man das ja verallgemeinern. Suchen Sie sich mal einen von beiden aus.

FW: Okay, dann nehme ich mal das $\vee$. Da muss man dann testen, ob mindestens eines der Kinder in einem q_1 Zustand ist. Also sowas wie $Q^* q_1 Q^*$ würde gehen.

WT: Sehr schön. Kommen wir mal zu unendlichen Systemen. Sagen wir mal wir haben einen Automaten mit einer FIFO-Queue. Was ist eine Konfiguration dieses Systems?

FW: Zustand plus Queueinhalt.

WT: Okay, dann ist jetzt auch klar, was ich mit Erreichbarkeit von Konfigurationen meine.

FW: Ja.

WT: Und? Ist das jetzt entscheidbar?

FW: Wenn die Queue unbeschränkt ist dann nicht. Man kann da eine Turing Reduktion machen. Und zwar kann der Automat dann durch die Queue scrollen und diese als Band für die Turing Maschine benutzen. Dann merkt man sich immer die letzten drei Zeichen. Wenn das alles Bandsymbole sind, scrollt man weiter, ansonsten macht man sein Update und dann ...

WT: Ja ja ok, das reicht. Was ist denn, wenn das Alphabet dieser Queue jetzt nur ein einziges Element hat?

FW: Hmm. Dann ist das im Grunde ja das gleiche wie ein Stack. Es spielt ja keine Rolle ob ich vorne oder hinten lese oder schreibe. Also, dann hat man ein Countersystem mit einer Variable. Da ist Erreichbarkeit entscheidbar.

WT: Warum?

FW: Weil das ein Spezialfall von Stacks ist, und da geht es mit dem Satz von Büchi.

WT: Was sagt der denn?

FW: Wenn man eine reguläre Menge von Konfigurationen $C \subseteq P\Gamma^*$ hat, dann ist auch die pre* und post* Menge regulär... Und die Automaten dafür kann man sogar konkret angeben!

WT: Was hat der Automat den man dann kriegt denn mit dem Automaten für C zutun?

FW: Naja, das ist eigentlich erstmal der selbe... *Prof. Thomas guckt skeptisch.* Also, nicht der selbe. Aber man nimmt den erstmal, und fügt dann Kanten ein.

WT: Okay, genau. Sagen wir mal ich habe jetzt eine Konfiguration. Wie kann ich testen ob man von hier aus unendlich viele weitere erreichen kann?

FW: Das ist leicht. Man baut den post* Automaten und guckt ob die Sprache unendlich ist.

WT: Wie soll das denn gehen?

FW: Das ist ein NEA. Man guckt, ob es einen Kreis im Graphen gibt.

WT: einen Kreis?

FW: der muss natürlich erreichbar sein und muss zu einem Endzustand führen.

WT: Ja genau. Und das geht effizient?

FW: Ja.

WT: Okay. Gut, kommen wir mal zu den unendlichen Wörtern.

Automaten auf unendlichen Wörtern

WT: Was ist eine Co-Büchi-Sprache?

FW: Das sind die Komplemente der Büchi-Sprachen, daher auch der Name. Man definiert das über eine andere Akzeptanzbedingung. Bei Büchi sagt man ja, dass man unendlich oft einen Endzustand sehen muss. Hier akzeptiert man jetzt, wenn man irgendwann keine Nicht-Endzustände mehr sieht.

WT: Ja, so könnte man das auch ausdrücken. Bei NP und Co-NP weiss man ja nicht ob die beiden verschieden sind. Wie es ist hier? Sind Büchi und Co-Büchi gleich?

FW: Nein, die sind unvergleichbar. Keine von beiden ist unter Komplement abgeschlossen.

WT: Gut, dann hatten wir ja noch Muller-Automaten. Können Sie mir einen Algorithmus sagen, der entscheidet ob mein Muller-Automat eine Sprache erkennt, die man auch Co-Büchi-erkennen könnte?

FW: Das geht mit dem Satz von Landweber. Die Co-Büchi-erkennbaren sind genau die, deren Automaten unter Unterschleifen abgeschlossen sind.

WT: Wie? Was soll abgeschlossen sein?

FW: Ähm, also, die Endzustandsmenge. Wenn man eine Schleife als Endzustand hat, dann muss auch jede kleinere Schleife einer sein. *Herr Thomas guckt etwas skeptisch.* Mit „kleiner“ meine ich natürlich Inklusion. *Herr Thomas freut sich.*

WT: Gut, um welchen Muller-Automaten geht es denn da mit den Schleifen? Woher krieg ich den?

FW: Das ist egal, das gilt für jeden beliebigen Muller-Automaten für die Sprache.

WT: Genau, sehr schön. Welche Richtung von diesem Beweis ist denn schöner?

FW: Die von den Schleifen zu dem Co-Büchi-Automaten. Da kippt man einfach alle Endzustandsmengen zusammen. Die andere Richtung ist aber auch nicht so schlimm.

So im Nachhinein bin ich mir gar nicht mehr sicher, ob das überhaupt stimmt. Ich habe das glaube ich mit E-erkennbaren Sprachen verwechselt. Hat Prof. Thomas aber scheinbar überhört.

WT: Naja, das ist Geschmackssache. Aber lassen wir das mal. Welche Logik ist äquivalent zu den Muller-Automaten?

FW: Also, bei endlichen Wörtern hat man ja MSO. Das kann man hier auch machen, nur dass die Wortstrukturen etwas anders aussehen, im wesentlichen ist das $(\omega, <)$. Das nennt man dann die S1S-Logik.

WT: Wie macht man denn jetzt aus einem Automaten eine Logik Formel?

FW: Das geht genauso wie bei endlichen Wörtern. Man sagt, dass es einen akzeptierenden Lauf gibt.

WT: Ok. Muller-Automaten sind ja äquivalent zu nicht-deterministischen Büchi Automaten. Wie zeigt man das?

FW: Die Struktur von solchen Wörtern ist ja immer, dass man irgendeinen endlichen Präfix hat und danach geht man in eine Schleife. Der Büchi Automat rät jetzt, wann der Präfix vorbei ist und rät in welches F_i man geht.

WT: Und wie geht das formal?

FW: Man sammelt die Zustände ein und resettet wenn man alle aus F_i gesehen hat. Endzustände sind die Reset-Zustände.

WT: Genau. Machen wir mal einen kleinen Schwung zu Model Checking.

Die Grenzen zwischen den Vorlesungen waren sehr schwammig und im Model Checking Teil hat er eigentlich nichts gefragt, was nicht auch irgendwie in den Skripten zu den anderen beiden Vorlesungen stand. Also ruhig weiter lesen :-)

Model Checking

WT: Es gibt da ja die Logik LTL. Was hat die mit S1S zutun?

FW: LTL ist äquivalent zu dem FO-Fragment von S1S.

WT: Was hat man da so für Operatoren in LTL?

FW: X, U, F, G

WT: Welche davon braucht man?

FW: X und U

WT: Schreiben Sie mal F oder G durch die anderen.

Ich versuche vergeblich G durch Until auszudrücken. Prof. Thomas sagt dann, ich soll es lieber mal F probieren. Dann ist es mir eingefallen.

FW: $F\varphi \equiv \texttt{true}U\varphi$. Und dann hat man $G\varphi \equiv \neg F \neg \varphi$

WT: S1S Model Checking ist ja jetzt leicht indem man den Automaten dazu macht. Wozu braucht man denn jetzt noch LTL?

Ohne zu Wissen was ich damit eigentlich genau sagen will, antworte ich direkt:

FW: Bei LTL hat man ein paar schönere algorithmische Eigenschaften.

WT: Ach so? Welche denn?

Mist. Hätte erstmal nachdenken sollen :-) Ich hatte im Gefühl, dass hier irgendwas entscheidbar ist was für S1S unentscheidbar ist. War natürlich Quatsch.

FW: Hmm. Also Model Checking, Erfüllbarkeit, Leerheit etc. sind bei beiden entscheidbar. Hmmm.

WT: Naja, es geht ja nicht immer nur darum **ob** es geht, sondern...?

FW: Wie schnell es geht!

WT: Genau. Wie geht das denn jetzt überhaupt bei LTL?

FW: Man will wissen ob $TS \models \varphi$, d.h. $\text{Traces}(TS) \subseteq \text{Words}(\varphi)$. Da kann man jetzt über Leerheit gehen: $\text{Traces}(TS) \cap \text{Words}(\neg\varphi) = \emptyset$. Die Inklusion gilt, wenn der Schnitt mit dem Komplement leer ist. Das Komplement ist blöd, da müsste man den Büchi-Automaten komplementieren, deswegen macht man das lieber auf Seite der Logik.

WT: Ja, und jetzt zur Geschwindigkeit. Wie schnell geht das? Wie groß wird der Büchi Automat für die Formel?

FW: Der wird exponentiell groß in der Anzahl der Teilformeln: $O(2^{|\varphi|})$.

Irgendwie sind wir jetzt auf NP*-Schwere gekommen.*

WT: Wie ist das mit P und NP? Wissen **Sie** ob die gleich sind?

FW: Hehe, da hat sich in den letzten Tagen einiges getan, aber ich weiss das nicht. [4]

WT: Ja ich bin auch nicht überzeugt. Zurück zum Thema. Wie ist es bei S1S?

FW: Hmm..

WT: Bei LTL ist es ja **einfach** exponentiell!

FW: Aaaah, ja genau. Da gibt es einen Satz. Bei endlichen Wörtern ist das der Satz von Meyer und Stockmeyer glaube ich. *Prof. Thomas hat sich sichtlich sehr gefreut, dass ich diesen Namen kannte!* Bei S1S gibt es diesen Satz auch. Der sagt, dass es kein k gibt, so dass die Konstruktion immer in k-facher Exponentiation geht.

WT: Genau! Deswegen ist LTL also schön. CTL hatten wir auch. Was ist der Unterschied?

FW: Bei LTL hat man in der Semantik ja immer implizit diese „für alle Pfade gilt, ...“-Sache drin. Bei CTL kann man das explizit machen bzw. kann auch sagen „es gilt für **einen** Pfad“.

WT: Okay. Was ist denn mit dieser Formel: $AGFp$. Was sagt die?

FW: Auf allen Pfaden gilt, dass immer wieder Zustände vorkommen, an denen p gilt.

WT: Ist das eine CTL Formel?

FW: Nein, das ist CTL*. In CTL muss vor jedem Temporaloperator ein Quantor stehen.

WT: Kann man daraus eine CTL Formel machen?

FW: Hmm. Also es gibt da einen Satz, der sagt, dass **wenn** es geht, man nur Quantoren dazwischen mogeln muss. Also sage ich mal AGAFp.

WT: Ja genau. Kommen wir mal zum symbolischen Model Checking. Sagt ihnen das was?

FW: Ja. Da geht es darum, dass man das sog. „State Space Explosion Problem“ in den Griff bekommen will. Man sucht kompakte Darstellungen für das Transitionssystem in Form von Boolschen Funktionen.

WT: Wofür macht man das? Für CTL oder LTL?

FW: CTL

WT: Genau. Was sind das jetzt für Darstellungen?

FW: Entscheidungsdiagramme. Sogenannte OBDDs. Die haben ein paar nette Eigenschaften, z.B. effiziente boolsche Verknüpfungen und Äquivalenztests.

WT: Aha, wie testet man denn Äquivalenz?

FW: Das ist wie bei DEA. Man minimiert und guckt ob man Isomorphie hat.

WT: Okay, wenn ich jetzt so eine Formel habe und mache ein OBDD daraus. Wie kann ich dann testen ob die Formel erfüllbar ist?

FW: Man guckt ob ein 1-Blatt erreichbar ist.

WT: Ja genau! Das ist aber doch ganz leicht, oder?

[4] Gemeint war natürlich der gerade kürzlich vorgestellte Beweisvorschlag von Vinay Deolalikar

Hier war mir schon klar, dass das auf den scheinbaren Widerspruch zur NP*-Vollständigkeit von SAT rausläuft.*

WT: Haben Sie den $P \neq$ NP Beweisvorschlag gelesen?

Ich sage ganz vorsichtig Ja, obwohl ich natürlich kein Wort verstanden habe und den Beweis nur überflogen habe.

WT: Was benutzt der da?

FW: Irgendeine $k-$SAT Reduktion.

WT: Genau. Das ist ja NP vollständig für $k \geq 3$. Sie haben aber gerade gesagt, dass das mit OBDDs einfach ist.

FW: Ja, aber das ist kein Widerspruch zu SAT. OBDDs können sehr groß werden.

WT: Was wollen Sie damit sagen?

FW: Naja, es gibt Formeln, sagen wir mal in KNF, für die jedes äquivalente OBDD exponentiell groß ist. Da geht das also wieder alles kaputt.

WT: Sehr schön. Gehen wir mal wieder zurück zur anderen Vorlesung. Wir hatten ja auch unendliche Spiele.

Unendliche Spiele

WT: Eben hatten wir Muller-Automaten. Jetzt machen wir mal Muller-Spiele. Wie löst man die denn?

FW: Also mit „Lösen“ meint man, dass man wissen möchte wer an welchen Startpositionen eine Gewinnstrategie hat. Bei Muller sieht man sofort, dass das nicht ohne Speicher geht. Man muss sich merken, was schon passiert ist. Man kann sogar zeigen, dass man manchmal $O(n!)$ Speicher braucht.

WT: Und wie kommt man jetzt an den Automaten?

FW: Indem man sich den Zusammenhang zu Paritätsspielen anguckt. Aus der Spielreduktion erhält man die Strategie.

WT: Wie sieht die Reduktion denn aus?

Ich war hier im Kopf schon dabei was vom Einsammeln von Zuständen zu erzählen als mir noch rechtzeitig eingefallen ist, dass man dafür LAR Records benutzt und ich das mit Staiger-Wagner verwechsle.

FW: Dafür benutzt man sog. „Latest Appearance Records“. Das ist eine Datenstruktur mit der man sich quasi merken kann, was unendlich oft passiert ist. Das sind solche Tupel von Zahlen bzw. Permutationen davon und eine Markierung. Dann gibt es das LAR Lemma, das sagt, dass wenn man eine Endzustandsmenge mit m Elementen hat, dass dann irgendwann nur noch Markierungen $< m$ vorkommen und das F dann als Hitsegment unendlich oft auftaucht.

WT: Und wie kommt man jetzt zu Paritätsspielen?

FW: Man färbt die Records nach ihrem Hitsegment. Die Farbe ist $2 \cdot h$ wenn das Hitsegment ein Endzustand ist und $2 \cdot h - 1$ wenn nicht.

WT: Genau. Schön. Wir hatten auch noch viel einfachere Spiele, wo man diese Automaten gar nicht braucht. Was ist das aller einfachste Spiel?

FW: Das sind Erreichbarkeitsspiele. Ich habe gewonnen, sobald ich eine bestimmte Position erreiche.

WT: Genau, wie löst man das?

FW: Das ist viel einfacher und geht ohne Gedächtnis. Man guckt sich einfach an von wo aus man den Gegner zum Endzustand hin zwingen kann. Das geht über die Attraktorkonstruktion.

WT: Wie genau?

FW: Naja man fängt mit den Endzuständen an und geht dann iterativ vor. Man fügt immer Knoten hinzu bei denen ich am Zug bin bzw. der Gegner, aber er keine andere Wahl hat. Das gibt dann so eine Inklusionskette. Die wird nach spätestens $|Q|$ Schritten stationär.

WT: Wie schnell geht das also?

FW: Hmm. Quadratisch in $|Q|$. *Ich war mir nicht sicher ob das stimmt und habe auch ziemlich rumgestammelt, als ich es erklären sollte.*

WT: Ja, genau. Ich sage jetzt das geht auch in $O(|Q| + |E|)$.

FW: Das ist auch quadratisch.

WT: Ja schon, aber „besser“ als Ihr quadratisch. *Grinst.* Wie könnte das gehen?

FW: Hmmm. *Minute überlegt.* Weiss ich nicht.

WT: Das war eine Übungsaufgabe. Kommt ihnen diese Komplexität nicht bekannt vor?

FW: Das sieht sehr nach Breitensuche aus.

WT: Genau!

War nicht schlimm, dass ich es nicht hinbekommen habe. Prof. Thomas hat es mir dann erklärt. Man macht zwei Breitensuchen. Rückwärts von F geht man durch den Graphen und zählt die Kanten, die in den Attraktor gehen und schreibt sie an die Knoten. Beim zweiten Durchlauf überprüft man ob das alle waren.

WT: Wie ist das jetzt hier? Gewinnt immer jemand?

FW: Ja, die Spiele sind determiniert, das heisst ...

WT: Jaja schon gut. Okay, nochmal zu den Paritätsspielen. Wie sieht es da aus?

FW: Die sind positional, d.h. man kommt ohne Gedächtnis aus.

WT: Wie schwer ist das denn?

FW: Was meinen Sie?

WT: Ich gebe Ihnen einen Knoten und will wissen wer da gewinnt.

FW: Das ist in NP. Und wegen der Symmetrie der Gewinnbedingung natürlich auch in Co-NP.

WT: Wie zeigt man das?

FW: Also, NP kann man ja klassifizieren als die Probleme, deren Lösungen sich effizient verifizieren lassen. Wenn ich jetzt eine positionale Strategie nehme, kann ich die ja einfach kodieren indem ich alle Kanten, die ich nicht nehmen möchte, rauslösche. Das ist also schonmal polynomiell. Jetzt teste ich ob es einen Kreis gibt auf dem die höchste Farbe ungerade ist.

WT: Einen Kreis? Ach so, das meinen Sie. Ja genau. Und das geht polynomiell?

FW: Ja.

WT: Nochmal zu Paritätsspielen. Da gibt es ein paar Verbindungen zu Baum-Automaten. Was wissen Sie darüber?

FW: Man kann Paritätsspiele benutzen um zu zeigen, dass eine Baumsprache unter Komplement abgeschlossen ist. Wir haben das im Baum-Satz und Basis-Satz von Rabin gebraucht.

WT: Ja bleiben wir mal beim Complementation Lemma. Wie geht das?

FW: Also, wenn der Baum nicht in der Sprache ist, dann gewinnt Spieler II und ...

WT: Moment mal. Was für Spieler? Um welches Spiel geht es überhaupt?

FW: Man kann zu jedem Baum-Automaten ein Spiel definieren. Einer der Spieler ist dann der Automat und der andere ist der „Pfadfinder“. Dann hatten wir das Run Lemma, das sagt, dass ein Baum genau dann in einer Sprache ist, wenn der Automat an der Wurzel gewinnt. Also, eigentlich ist die Negation viel wichtiger. Wenn der Baum **nicht** drin ist, also, im Komplement ist, dann gewinnt der Pfadfinder. Hier braucht man jetzt die Spieltheorie. Die Spiele sind positional determiniert. Und aus der Strategie von Spieler II macht man jetzt die Automatenstrategie für so einen anderen Muller-Automaten.

WT: Ja genau, für einen anderen. *Prof. Thomas schien das wichtig zu finden, dass ich gesagt habe, dass es ein* **anderer** *zweiter Automat ist. Vermutlich wird das in Prüfungen oft falsch erklärt.*

WT: Gut, wir sind dann hier fertig. Bitte warten Sie draußen.

Fazit

Tolle Prüfung. Das war eigentlich keine Prüfung sondern ein nettes kleines Gespräch mit Prof. Thomas über den Stoff. Sehr entspannt, Prof. Thomas ist sehr nett und ruhig, lässt einen nachdenken und erklärt alles, was man nicht weiß, so dass man sogar noch was in der Prüfung lernt. Er hat mir hinterher gesagt, dass er es sehr schön fand, dass ich so schnell und so viel geredet habe und dass er es schrecklich findet wenn ein Prüfling sich alles aus der Nase ziehen lässt. Meine paar kleineren Fehler waren nicht tragisch. Er fand es sehr schön, dass ich Fragen beantworten konnte, die nie in der Vorlesung vorkamen. Ich kann Prof. Thomas als Prüfer uneingeschränkt empfehlen.

Es hat mich etwas gewundert, dass er fast überhaupt nicht in die Tiefe gefragt hat sondern nur sehr in die Breite. Ich musste nichts formal beweisen, nicht einmal formal definieren. Die meiste Zeit der Prüfung war mündlich und ich musste, im Vergleich zu meiner Prüfung in Theoretischer Informatik oder im Nebenfach Mathematik, sehr wenig aufschreiben.

Zu Model Checking kann ich sagen, dass die Vorlesung sehr gut aus dem Buch von Prof. Katoen lernbar ist. Das Buch ist unglaublich ausführlich, fast schon zu ausführlich. Wie erwartet hat Prof. Thomas hier am wenigsten gefragt und auch am wenigsten in die Tiefe. Alles was er gefragt hat, hätte man eigentlich auch aus seinen eigenen Skripten wissen können.

Gelernt habe ich etwa 5 Wochen lang in einer Zweiergruppe.