Gedächtnisprotokoll Diplomprüfung Vertiefungsgebiet Automaten, Logik und Verifikation

Prüfling: Michael Holtmann (Michael.Holtmann@rwth-aachen.de)
Prüfer: Prof. Dr. Dr.h.c. Wolfgang Thomas
Beisitz und Protokollführung: Jan-Henrik Altenbernd
Datum: 11.04.2006, 9:30 Uhr
Geprüfte Fächer:

1. Automata and Reactive Systems (Vorlesung, Wintersemester 2002/2003)
2. Angewandte Automatentheorie (Vorlesung, Sommersemester 2003)
3. Model Checking (Vorlesung Katoen, Wintersemester 2005/2006)

Dauer: ca. 45 Minuten
Note: 1.0

1.) Automata and Reactive Systems
WT: Wir haben ja die Büchi-Automaten kennengelernt. Wann akzeptiert denn so ein Büchi-Automat?
Ich: Wir haben eine Endzustandsmenge F gegeben und der Automat akzeptiert, wenn die Menge der unendlich oft gesehenen Zustände mit dieser Menge einen nicht-leeren Schnitt hat. Man muss also unendlich oft durch einen Endzustand kommen.
WT: Wenn wir jetzt mal nur die deterministischen Automaten nehmen. Da gibt es ja auch noch die co-Büchi-Automaten. Was ist denn da der Unterschied zum Büchi-Automaten...?
Ich: Wie gesagt, der Büchi-Automat akzeptiert, wenn wir unendlich oft einen Endzustand sehen. Der co-Büchi-Automat akzeptiert, wenn wir von einem bestimmten Zeitpunkt an bis in die Unendlichkeit nur noch Endzustände sehen.
WT: Kennen sie ein Ergebnis aus der Vorlesung, dass die beiden irgendwie in Verbindung bringt?
Ich: Ja, da haben wir das Komplement-Lemma gehabt. Das sagt, dass eine Sprache deterministisch Büchi-erkennbar ist genau dann, wenn ihr Komplement deterministisch co-Büchi-erkennbar ist.
WT: Wir haben da ja auch noch Muller-Automaten behandelt, sowohl deterministische als auch nichtdeterministische. Unterscheiden die sich denn in

der Ausdrucksstärke?
Ich: Deterministische und nichtdeterministische Muller-Automaten sind gleich ausdrucksstark.
WT: Richtig. Wenn wir die Muller-Automaten jetzt mal in Verbindung bringen mit den Büchi-Automaten und da auch mal den nichtdeterministischen Fall betrachten. Sind die Modelle äquivalent?
Ich: Also, die Muller-Automaten sind gleich ausdrucksstark wie nichtdeterministische Büchi-Automaten…
WT:…wie sieht es denn da mit den Abschluss-Eigenschaften aus?
Ich: Nichtdeterministische Büchi-Automaten und Muller-Automaten sind abgeschlossen unter Komplement. Aber deterministische Büchi-Automaten sind es nicht.
WT: Können Sie mir eine Sprache nennen, die man nicht durch einen deterministischen Büchi-Automaten erkenn kann?
Ich: Ja, die Sprache, wo ab irgendeinem Zeitpunkt nur noch ein Buchstabe zugelassen ist, wenn wir z.B ein Alphabet mit zwei Buchstaben haben. (schreibe auf: $(0 + 1)^*1^\omega$)
WT: Wie kann man das denn beweisen?
Ich: Naja, betrachten wir erst mal das Wort 1^ω. Das ist ja in der Sprache. Also muss ein deterministischer Büchi-Automat darauf unendlich oft durch einen Endzustand kommen. Und wenn er unendlich oft durch einen Endzustand kommt, dann kommt er auch nach einem endlichen Präfix durch einen Endzustand, sagen wir nach n_0 Einsen (schreibe auf 1^{n_0}). Dann hängt man eine 0 dran...(schreibe 0 dazu). Dann macht man wieder mit Einsen weiter…
WT:…Ja, ok. Das reicht schon. Es ist klar.
Ich: Ja, man wiederholt das immer und immer wieder.
WT: OK. Kennen Sie denn eine Möglichkeit, wie man entscheiden kann, ob eine Sprache deterministisch Büchi-erkennbar ist?
Ich: Ja. Dazu haben wir den Satz von Landweber in der Vorlesung gehabt. Wennn wir einen Muller-Automaten gegeben haben, der die Sprache erkennt, dann können wir anhand der Akzeptier-Komponente $\mathcal{F}$ dieses Automaten feststellen, ob die Sprache deterministisch Büchi-erkennbar ist. Die Sprache ist deterministisch Büchi-erkennbar genau dann, wenn die Akzeptierkomponente aus diesen F-Mengen abgeschlossen ist unter Oberschleifen.
WT: Dann machen sie das doch mal für diese Sprache.
Ich: (schreibe einen Automaten mit zwei Zuständen 1 und 2 auf) Bei einer 0 gehen wir immer in den Zustand 1 (füge Transitionen von Zustand 1 mit 0 nach 1 und von Zustand 2 mit 0 nach 1 ein)…und bei einer 1 gehen wir immer nach Zustand 2 (füge Transitionen von Zustand 1 mit 1 nach 2 und von Zustand 2 mit 1 nach 2 ein)…(schreibe auf: $\{\{2\}\}$) die Akzeptiermenge besteht dann hier nur aus einer Menge mit Zustand 2.

WT: Ja, und jetzt sieht man es wohl schon...
Ich: Ja, genau. Die Menge $\{1,2\}$ ist eine Oberschleife von $\{2\}$, ist aber nicht in $\mathcal{F}$ enthalten. Also ist $\mathcal{F}$ nicht abgeschlossen unter Oberschleifen.
WT: OK, gut. Büchi-Automaten haben wir ja auch im Model Checking behandelt, aber das machen wir dann später...Sie haben gesagt, dass Muller-Automaten gleich ausdrucksstark wie nichtdeterministische Büchi-Automaten sind. Können Sie mir mal die Idee für beide Richtungen des Beweises erläutern?
Ich: Nun, die leichtere Richtung ist mit Sicherheit die von Muller nach Büchi. Da konstruiert man einen nichtdeterministischen Büchi-Automaten der zwei Dinge rät. Zum einen diejenige Menge F_i, die unendlich oft gesehen wird und zum anderen die Position im Lauf, aber der genau diese Menge besucht wird. Jedes Mal, wenn das F_i erreicht wird, wird die Mengenkomponente zurückgesetzt auf die leere Menge und damit ein Endzustand erreicht.
WT: Richtig. Und wie sieht es da mit der Komplexität aus. Ist das polynomiell?
Ich: Nein, das ist exponentiell. Man merkt sich ja immer diese Menge F_i.
WT: Und wie sieht nun die andere Richtung aus?
Ich: Bei der anderen Richtung haben wir die Safra-Konstruktion kennengelernt.
WT: Richtig. Wie sieht es denn aus, wenn wir die Komplexitäten der beiden Richtungen vergleichen. Sie sagen das eine ist exponentiell...da verstehe ich mal 2^n drunter...Geht denn die Safra-Konstruktion auch in dieser Zeit?
Ich: Nein. (schreibe auf: $2^{O(n \cdot log(n))}$) Die Komplexität der Safra-Konstruktion ist $2^{O(n \cdot log(n))}$.
WT: Und geht das denn besser?
Ich: Hmmm...ich weiß, dass es da ein ganz neues Ergebnis zu gibt, dass es bei Muller-Automaten wohl nicht besser geht.
WT: Wirklich. (schaut verblüfft) Von wem ist das denn?
Ich: Also, von wem das ist weiß ich nicht. Das hat mir nur Herr Löding erzählt. Der musste zu dem Paper wohl ein Gutachten schreiben und meinte, das sähe schon richtig aus.
WT: Achso, ja von **irgendein Name**. Für mich ist das immer noch offen, weil ich mir den Beweis noch nicht angesehen habe. Aber wie haben wir das denn für Rabin-Automaten gezeigt?
Ich: (überlege...nach etwas Nachdenken und Denkanstößen vom Prof)...also, wir können einen Rabin-Automaten konstruieren. Der hat nur linear viele akzeptierende Paare. Jedes Paar erfasst exponentiell viele Muller-Mengen, aber die Anzahl der Zustände...die geht nicht besser, meine ich.
WT: Wie kommen sie denn darauf?
Ich: (war mir dann ganz kurz nicht sicher, wie genau die Schranke war, die wir gezeigt hatten. Aber dann kam ich wieder drauf) Wir haben für Rabin-

Automaten gezeigt, dass es nicht mit $2^{O(n)}$ Zuständen geht.
WT: Woran liegt das denn?
Ich: Dazu haben wir das Union Lemma benutzt (als das Wort "Union-Lemma" fiel, schaute er zufrieden). Die Vereinigung der Inf-Mengen von zwei nicht-akzeptierenden Läufen ist wieder nicht-akzeptierend. Darüber haben wir gezeigt, dass mindestens $2^{O(n \cdot log(n))}$ Zustände nötig sind.
WT: Gut, gehen wir mal weiter zu Baumautomaten. Wir haben zum Beispiel Büchi-Baumautomaten und Rabin-Baumautomaten kennengelernt.
Ich: Naja, wir haben meist Paritäts-Baumautomaten betrachtet...
WT: Ja, stimmt. Wenn wir das jetzt mal mit dem Fall der Wörter vergleichen. Können wir bei Bäumen denn nicht auch mit Büchi-Automaten arbeiten?
Ich: Nein, denn über Bäumen sind Büchi-Automaten echt schwächer als Paritätsautomaten.
WT: Woran liegt das denn?
Ich: Die Sprache der Bäume mit nur endlich vielen b auf jedem Pfad ist nicht durch einen nichtdeterministischen Büchi-Baumautomaten erkennbar.
WT: Wie kann man das denn zeigen?
Ich: Wir haben dazu einen Baum konstruiert, der zwar unendlich viele b enthält, aber nur endlich viele auf jedem Pfad.
WT: (wiederholt meine Worte mit skeptischem Blick)...mit unendlich vielen b?!?
Ich: Ja, das haben wir so gemacht.
WT: Ja stimmt...mit unendlich vielen b. Und wie geht dann der Beweis?
Ich: Wir haben gezeigt, dass man aus diesem Baum einen anderen Baum konstruieren kann, der einen Pfad mit unendlich vielen b hat, so dass auf diesem Baum ein akzeptierender Lauf existiert. Auf dem kritischen Pfad wiederholt sich ein akzeptierender Zustand und dazwischen kommt mindestens ein b vor. Diesen Teil des Baumes können wir dann wiederholt darunter kopieren.
WT: (da schaute er schon zufreiden) Ja, genau. Die entscheidene Idee ist dieses wiederholte Kopieren des Teils mit dem b drin.
Ich: Dadurch haben wir dann einen Pfad auf dem unendlich viele b vorkommen. Der Baum wird aber auch akzeptiert.
WT: Gut. Kommen wir mal zu der Verbindung zur Logik. Zu welcher Logik sind denn Paritäts-Baumautomaten äquivalent?
Ich: Das ist $S2S$.
WT: Richtig. Wofür steht denn die 2 in $S2S$?
Ich: Das heißt, dass es zwei Nachfolger gibt.
WT: Wie beweisen wir denn die beiden Richtungen für die Äquivalenz von Paritäts-Baumautomaten und $S2S$?
Ich: Das ist ein Ansatz, den wir analog auch in anderen Gebieten verwendet

haben. Wenn wir von den Automaten zur Logik gehen, dann drücken wir aus, dass ein erfolgreicher Lauf existiert.
WT: Wie sieht denn die Formel dafür aus?
Ich: Wir müssen zum einen ausdrücken, dass eine Partition existiert, so dass zu jedem Zeitpunkt nur ein Zustand angenommen werden kann...
WT: Ja, ok...aber ich meinte eher die Art der Formel. Welche Quantoren werden denn verwendet?
Ich: Achso, das ist eine existentielle Formel.
WT: Was muss denn dort im Inneren alles ausgedrückt werden...
Ich:...ach, das war ja falsch, was ich gesagt habe. Wir verwenden im Inneren der Formel ja noch den $\forall$-Quantor. Damit quantifizieren wir über alle existierenden Pfade, um auszudrücken, was auf den Pfaden gelten muss.
WT: Ja, genau. Das nennt man auch eine Σ_2-Formel.
Ich: Ach, richtig. Das haben Sie mal in der Vorlesung erwähnt.
WT: Wie funktioniert denn die andere Richtung?
Ich: Da gehen wir induktiv über den Formelaufbau. Wir reduzieren $S2S$ zunächst auf $S2S_0$ und konstruieren dann Automaten für die elementaren Formeln. Im Induktionsschritt nutzen wir die Abschlusseigenschaften von Paritäts-Baumautomaten aus.
WT: Welche brauchen wir dann?
Ich: Das sind die Disjunktion, die Negation und die existentielle Quantifizierung.
WT: Was kosten die denn so?
Ich: Die Disjunktion ist einfach, da verwenden wir einfach den Vereinigungsautomaten. Die Negation ist teuer, da haben wir ja diese komplizierte Komplementierung von Paritäts-Baumautomaten betrachtet.
WT: Und beim $\exists$-Quantor?
Ich: Das ist auch billig. Da vergessen wir einfach die letzte Komponente der Transitionen.
WT: Wo haben wir denn Paritäts-Baumautomaten eingesetzt?
Ich: Zum Beispiel im Beweis zu Rabins Baum-Satz.
WT: Was sagt der denn aus?
Ich: Der Satz sagt aus, dass die monadische Theorie des binären Baumes entscheidbar ist.
WT: Und wie haben wir das gezeigt?
Ich: Wir haben aus einem gegebenem $S2S$-Satz einen Paritäts-Baumautomaten ohne Eingabe konstruiert und die von ihm erkannte Sprache dann auf Leerheit überprüft. Das geht nach dem Basis-Satz von Rabin. Letzterer wird mit Hilfe von Paritätsspielen bewiesen.
WT: Wie funktioniert der Beweis denn genau?
Ich: Von einem gegebenen Baumautomaten gehen wir über zu einem Baum-

automaten ohne Eingabe. Dieser rät einen Baum und simuliert den Lauf des ursprünglichen Baumautomaten auf diesem Baum. Nach dem Run Lemma hat dieser Baumautomat einen erfolgreichen Lauf genau dann, wenn Spieler Automaton eine Gewinnstrategie im zugehörigen Paritätsspiel hat. Das können wir feststellen, denn zur Lösung von Paritätsspielen auf endlichen Graphen haben wir Algorithmen zur Verfügung.
WT: Was sagt denn nun der Basis-Satz eigentlich genau?
Ich: Der Satz sagt, dass der Baumautomat eine nicht-leere Baumsprache erkennt genau dann, wenn Spieler Automaton eine Gewinnstrategie im zugehörigen Paritätsspiel hat.
WT: Ja, aber was folgt daraus? Wieso heißt der Satz denn "Basis-Satz"?
Ich: Achja, der Satz sagt, dass jede nicht-leere Baumsprache, die durch einen Paritäts-Baumautomaten erkannt wird, mindestens einen regulären Baum enthält. Der wird gerade durch den deterministischen Baumautomaten generiert, den die Gewinnstrategie von Automaton hervorbringt.
WT: Ja, genau. Gut, was ist denn, wenn man statt zwei Nachfolgern auch mehr Nachfolger zulässt.
Ich: Dann geht man über zu SnS.
WT: Wo können wir das denn gebrauchen?
Ich: ...zum Beispiel bei PDL.
WT: Richtig. Und wenn man beliebigen aber endlichen Verzweigungsgrad zulässt?
Ich: Ja, dann...
WT: Welche Baumautomaten haben wir denn da kennengelernt?
Ich: Das waren die XML-Automaten über unbeschränkt verzweigenden Bäumen. Aber die haben wir nur für den endlichen Fall behandelt.
WT: Ja, gehen wir mal zu Angewandte Automatentheorie...

2.) Angewandte Automatentheorie
WT:...da haben wir ja MSO kennengelernt als zu regulären Sprachen äquivalente Logik. Was ist denn mit der $<$-Relation. Brauchen wir die?
Ich: Nein, die können wir in MSO eliminieren.
WT: Wie lautet die zugehörige Formel.
Ich: Wir drücken aus, dass es eine Menge gibt, die das größere Element enthält und abgeschlossen ist unter Nachfolgern und das kleinere Element nicht enthält.
WT: Ja, gut. So kann man das auch machen. Dann heißt das also, dass man von dem einen Element das andere durch eine unter Nachfolgern abgeschlossene Menge nicht erreichen kann. Wenn wir das jetzt mal verallgemeinern, dann haben wir das...
Ich:...Erreichbarkeitsproblem.

WT: Richtig. Kennen Sie denn Automatenmodelle, über denen wir dieses Problem entscheiden können?
Ich: Ja, z.B. die Pushdownsysteme. Da haben wir zwei Saturierungsalgorithmen für $pre^*(C)$ und $post^*(C)$ kennengelernt.
WT: Da gibt es doch einen Satz. . . was sagt der denn?
Ich: Einen Satz?
WT: Ja, der Satz von Büchi. Der sagt etwas über dieses Problem aus. . .
Ich: Achso, ja. Wenn wir einen P-Automaten für C gegeben haben, dann können wir daraus effektiv, mit den Saturierungsalgorithmen, Automaten für $pre^*(C)$ und $post^*(C)$ bestimmen.
WT: Und was heißt das dann für diese Mengen C, $pre^*(C)$ und $post^*(C)$?
Ich:. . . (überleg). . . achso, ja. Das bedeutet, dass wenn C regulär ist, dass dann auch $pre^*(C)$ und $post^*(C)$ regulär sind.
WT: (er wollte wohl das Wort „regulär“ hören) Ja, genau.
WT: Können Sie mir die Komplexität dieser Saturierungsalgorithmen sagen?
Ich: Ja. Die laufen in polynomieller Zeit. Also, wir können nur für jedes Zustandspaar des gegebenen P-Automaten und jedes Symbol aus Γ eine Transition einfügen. Also erhalten wir $|Q|^2 \cdot \Gamma$, wobei das Γ weggelassen wird, wenn wir das Ganze in der O-Notation schreiben. . .
WT:. . . ja, richtig. Der Zeitaufwand ist also quadratisch in der Anzahl der Zustände. Gut, das reicht schon. Wo ist das Erriechbarkeitsproblem denn noch entscheidbar?
Ich: (nach einigem Überlegen) Also, ich könnte Ihnen einige Unentscheidbarkeitsergebnisse nennen. . .
WT: Ok. Welche Modelle haben wir denn kennengelernt, bei denen das Erreichbarkeitsproblem unentscheidbar ist?
Ich: Zum Beispiel das unendliche Gitter. Da haben wir eine Reduktion vom Halteproblem für Turingmaschinen betrachtet. Das hatten wir allerdings in Automata and Reactive Systems. In Angewandte Automatentheorie können wir aber für T-Systeme die Unentscheidbarkeit des Leerheitsproblems auf ähnliche Weise zeigen.
WT: Richtig. Kennen Sie noch ein anderes Automatenmodell, bei dem das Erreichbarkeitsproblem unentscheidbar ist?
Ich: Da hatten wir die CFSMs. Man zeigt das wieder durch eine Reduktion vom Halteproblem für Turingmaschinen.
WT: Woran liegt das denn, dass das Erreichbarkeitsproblem bei CFSMs unentscheidbar ist? Wo liegt der entscheidene Unterschied zu Pushdownsystemen?
Ich: Bei CFSMs haben wir eine Queue, bei Pushdownsystemen nur einen Stack. Wenn wir bei Pushdownsystemen auf ein bestimmtes Symbol im Kel-

ler zugreifen wollen, dann müssen wir endlich viel Information, die darüber liegt, abtragen. Wir können diese Information natürlich in den Zustand reinkodieren, aber das geht eben nur für endlich viele verschiedene Möglichkeiten. Bei den CFSMs können wir jedes beliebige Element in der Queue erreichen, indem wir es einfach an den Wortanfang schaufeln. Dazu nehmen wir vorne ein Symbol weg und fügen es hinten direkt wieder an, solange, bis wir das entsprechende Symbol erreichen. Da verlieren wir also keinerlei Information. CFSMs sind deswegen viel mächtiger als Pushdownsysteme.
WT: Richtig. Nehmen wir mal zwei andere Varianten her. Ich biete Ihnen was an und Sie sagen mir, ob das Erreichbarkeitsproblem entscheidbar ist. (schreibt auf: $u\cancel{v} \rightarrow uw$) Anstatt am Anfang des Wortes streichen wir am Ende des Wortes ein paar Buchstaben und ersetzten diese durch ein neues Wort w. Ist das Erreichbarkeitsproblem dann entscheidbar?
Ich: Ja. Das ist ja analog zum PDS. Statt einem Präfixersetzungssystem haben wir ein Suffixersetzungssystem.
WT: Gut. (schreibt auf: $\cancel{u_1}v\cancel{w_1} \rightarrow u_2vw_2$) Jetzt streichen wir sowohl vorne als auch hinten etwas und ersetzen das jeweils durch ein neues Wort…
Ich: (zögere etwas) Also, wir haben doch in der Vorlesung die Bemerkung gemacht, dass bei Infix-Ersetzungssystemen das Erreichbarkeitsproblem unentscheidbar ist, weil wir damit ja eine Turingmaschine simulieren können…
WT: …richtig…
Ich:…und ich meine, wir hätten das auch für Präfix-Suffix-Ersetzungssysteme gesagt, haben es aber nicht gezeigt. (An der Stelle war ich mir unsicher, weil ich die ganz Zeit überlegt habe, wie man einen Schritt einer Turingmaschine mit so einem Präfix-Suffix-Ersetzungssystem simulieren kann; ich habe dann etwas vor mich hin gemurmelt ohne recht weiter zu kommen.)
WT:…naja, aber das können sie ja auch gar nicht wissen…das ist das Promotionsthema von Herrn Altenbernd.
Ich: (Erleichterung! Erst nach der Prüfung fiel mir ein, wie man mit so einem Präfix-Suffix-Ersetzungssystem eine Turingmaschine ganz einfach simulieren kann: Wenn man am Wortanfang a durch ϵ und am Wortende ϵ durch a ersetzt, dann hat man nichts anderes getan, als das a vorne zu streichen und hinten wieder anzufügen. Damit kann man dann die Queue einer CFSM simulieren und hat demnach wieder ein unentscheidbares Erreichbarkeitsproblem.)
WT: Wie würden sie denn allgemein so ein Erreichbarkeitsproblem definieren?
Ich:…
WT: (zeichnet Grundbereich und Nachfolgerrelation einer Wortstruktur auf)
Ich:…
WT: Naja, für die Erreichbarkeit von y von x aus müssen Sie ausdrücken,

dass eine Menge existiert, die x enthält, abgeschlossen unter Nachfolgern ist und auch y enthält. Gut, Herr Holtmann. Dann gehen wir noch zu Petrinetzen. Was braucht man denn dort, wenn man Sprachen akzeptieren will?
Ich: Man definiert eine Menge von Anfangsmarkierungen und eine Menge von Endmarkierungen. Außerdem benötigt man noch eine Beschriftung der Transitionen, also eine Abblidung der Menge T nach Σ. Ein Wort w wird akzeptiert genau dann, wenn es eine Schaltfolge von Transitionen gibt, die von einer der Anfangsmarkierungen zu einer der Endmarkierungen führt, so dass die Folge der Beschriftungen der geschalteten Transitionen gerade das Wort w ist.
WT: Können Sie mir sagen, welche Sprachen man mit Petri-Netzen auf diese Weise erkennen kann und welche nicht.
Ich: (zeichne das wohl bekannte Inklusionsdiagramm auf) Also zunächst mal ist jede reguläre Sprache auch Petrinetz-erkennbar. . .Petrinetze können sich gut Anzahlen von Buchstaben merken, aber bei der Reihenfolge haben sie Probleme. Die Sprache der Palindrome gerader Länge können sie zum Beispiel nicht erkennen.
WT: Wie ist denn die Idee zum Beweis?
Ich: Also, die Anzahl der Markierungen, die von einer gegebenen Anfangsmarkeirung erreichbar ist, ist polynomiell in der Länge der Worte, die das Petrinetz liest. . .ich glaube, das war $(s + r)^r$, wobei r die Anzahl der Transitionen eines gegebenen Petrinetzes und damit fest ist. Die Anzahl der Wörter der Länge ist aber exponentiell in $|s|$, z.B. 2^s, wenn wir ein Alphabet mit zwei Buchstaben haben. Dann ist klar, dass wir ein Wort s finden können, so dass die Anzahl der verschiedenen Wörter mit einer Länge $|s|$ die Anzahl der erreichbaren Markierungen übersteigt.
WT: Ja, genau. Wir gehen jetzt ins Model Checking. . .

3.) Model Checking

WT:. . .Sie haben da ja verschiedene Eigenschaften kennengelernt, wie z.B. Sicherheits- und Lebendigkeitseigenschaften. Definieren Sie mal eine Sicherheitseigenschaft. Sie können Ihre eigene Formulierung wählen.
Ich: Eine LT-Eigenschaft heißt Sicherheitseigenschaft, wenn zu jedem Wort aus dem Komplement der LT-Eigenschaft ein Präfix dieses Wortes existiert, so dass keine Verlängerung dieses Präfix in der LT-Eigenschaft enthalten ist. Ein solches Präfix heißt schlechtes Präfix.
WT: Genau. Wie könnte man das denn noch formulieren?
Ich: Naja, dass es keine schlechten Präfixe geben darf.
WT: Ja, und wenn man es noch anders ausdrückt. Was muss dennn gelten?
Ich: $\ldots closure(P) = P$?!?$\ldots$
WT: Bleiben Sie mal bei Präfixen.

Ich: . . .
WT: Naja, es darf nur gute Präfixe geben. Was ist denn eine Lebendigkeitseigenschaft?
Ich: (schreibe auf: $pref(P) = (2^{AP})^*$) Eine LT-Eigenschaft heißt Lebendigkeitseigenschaft, wenn die endlichen Präfixe der Eigenschaft die Menge aller endlichen Worte ist.
WT: Und was bedeutet das?
Ich: Das bedeutet, dass wir für jedes endliche Wort ein unendliches Wort finden können, so dass wir, wenn wir es hinten an das endliche Wort dranhängen, in P landen.
WT: Und gibt es Eigenschaften, die sowohl Sicherheits- als auch Lebendigkeitseigenschaften sind?
Ich: Ja, eine. Das ist die LT-Eigenschaft, die genau alle unendlichen Wörter enthält.
WT: Kann man LT-Eigenschaften als Sicherheits- oder Lebendigkeitseigenschaften ausdrücken?
Ich: Ja, jede LT-Eigenschaft kann dargestellt werden als der Schnitt einer Sicherheits- und einer Lebendigkeitseigenschaft.
WT: Wenn wir jetzt eine Metrik auf der Menge der unendlichen Worte einführen, dann kriegen wir daraus einen Cantor-Raum. Das Thema hatten wir ja zumindest für Automaten und Reaktive Systeme ausgeklammert. . .
Ich:. . .fragen Sie ruhig. . .
WT: Wie kann man denn LT-Eigenschaften in diesem Kontext auffassen?
Ich: (mir fiel ein, dass im Skript stand, dass man jede LT-Eigenschaft als den Schnitt aus zwei Mengen darstellen kann. Das zweite war eine dichte Menge, nur bei der ersten war ich mir nicht mehr sicher) Also, man kann jede LT-Eigenschaft darstellen als den Schnitt. . .einer geschlossenen Menge, glaube ich. . .und einer dichten Menge.
WT: Das stimmt aber nicht so ganz. . .
Ich: (mir fiel es nicht mehr ein; habe dann noch irgendwas mit den „abgeschloffenen“ Mengen gesagt). . .
WT: Naja, die Sicherheitseigenschaften sind die geschlosssenen Mengen und die Lebendigkeitseigenschaften sind die dichten Mengen.
Ich: Ach, so war das.
WT: Kommen wir mal zum eigentlichen Model Checking? Wie funktioniert denn CTL-Model Checking?
Ich: Da gehen wir induktiv über den Formelaufbau vor. Man betrachtet den Syntaxbaum der Formel und berechnet für jede Teilformel die Menge der Knoten, an denen die jeweilige Teilformel gilt. Das geht effizient.
WT: Wie ist denn die genaue Komplexität?
Ich: (schreibe hin: $O((N + K) \cdot |\Phi|)$) Die Komplexität des Verfahrens ist

$O((N+K) \cdot |\Phi|)$. Für jede Teilformel müssen wir eine Tiefensuche durch das gegebene Transitionssystem durchführen.
WT: Ja, im schlechtesten Fall...
Ich: ...stimmt, im schlechtesten Fall.
WT: Wie sieht das denn aus, wenn wir eine Konjunktion als Teilformel gegeben haben?
Ich: Naja, die können wir direkt an jedem Knoten behandeln.
WT: ...und beim $\bigcirc$-Operator?
Ich: Da müssen wir alle Nachfolger betrachten.
WT: Genau. Wie sieht es denn mit der Logik LTL aus?
Ich: Da geht Model Checking nur in exponentieller Zeit. LTL-Model Checking ist PSPACE-hart. (zeichne das Transitionssystem aus den Diamanten auf, mit dem wir die Reduktion vom Halteproblem für Turingmaschinen durchgeführt haben) Da haben wir eine Reduktion vom Halteproblem für Turingmaschinen verwendet. Jeder Diamant kodiert den Inhalt einer Zelle auf dem Arbeitsband.
WT: Richtig! Welche Operatoren werden denn in der Formel verwendet, die die Reduktion liefert?
Ich: Es wird sehr oft der $\bigcirc$-Operator verwendet, um auszudrücken, was an den einzelnen Zuständen eines Pfades durch das Transitionssystem gelten muss. Der $\Box$-Operator wird auch verwendet. (schreibe hin: $\Box(begin \rightarrow \ldots)$) Es gibt eine Teilformel der Art $\Box(begin \rightarrow \ldots)$. (zeige auf den Anfangszustand des Transitionssystems) Der Anfangszustand des Transitionssystems bekommt das Label *begin*.
WT:...was ist denn mit dem $\Diamond$-Operator?
Ich:...der wird auch verwendet. Es wird ausgedrückt, dass irgendwann ein Endzustand erreicht wird.
WT: Genau.
Ich:...man muss zum Model Checking von LTL und CTL allerdings noch eine wichtige Bemerkung machen...
WT: ...ja?...
Ich: Naja, man sollte daraus nicht schließen, dass CTL-Model Checking schneller ist als LTL-Model Checking, denn LTL-Formeln können exponentiell kürzer sein als äquivalente CTL-Formeln. Zum Beispiel kann man das Problem des Hamiltonschen Pfades mit einer LTL-Formel polynomieller Länge ausdrücken. Das geht aber in CTL nur mit einer Formel, die exponentielle Länge hat...und wenn $P \neq NP$ gilt, geht das in CTL auch nicht besser.
WT: Ja, das stimmt. Ein sehr großes Problem ist auch die Explosion der Zustandsräume, die man betrachtet. Wie sieht es denn mit der Ausdrucksstärke von LTL und CTL aus?
Ich: LTL und CTL sind unvergleichbar. (schreibe hin: $\Diamond\Box a$) Wenn man bei-

spielsweise die Persistenzeigenschaft $\Diamond\Box a$ betrachtet. Die kann man in LTL ausdrücken, aber nicht in CTL. Wenn man ausdrücken möchte, dass ein Fehler in einem System immer behoben werden kann...das kann man in CTL ausdrücken, aber nicht in LTL. (schreibe hin: $\forall\Box\exists\Diamond a$) a drückt hier aus, dass ein Reset-Zustand erreicht wird. CTL* umfasst sowohl LTL als auch CTL. Wenn man z.B. die Oder-Verknüpfung dieser beiden Formeln (ich weise auf die beiden Beispielformeln hin) nimmt, dann hat man eine CTL*-Formel, die aber weder eine LTL- noch eine CTL-Formel ist.
WT: Reden wir mal über Bisimulationen. Wie sieht denn so eine Bisimulation aus?
Ich: Eine Bisimulation ist eine Menge von Zustandspaaren...(und dann etwas sehr umgangssprachlich)...Für jedes Paar in der Bisimulation muss für jede Kante die vom ersten Knoten des Paares ausgeht eine gleichbeschriftete Kante vom zweiten Knoten des Paares ausgehen und das aus den beiden Zielknoten resultierende Paar wieder in der Bisimulation sein. Das mit den ausgehenden Kanten muss auch umgekehrt gelten.
WT: Richtig! Wie sieht das mit bisimilaren Transitionssystemen im Bezug auf Formeln aus?
Ich: Bisimilare Transitionssysteme haben genau die gleichen Spuren und es gelten genau die gleichen CTL-Formeln und genau die gleichen CTL*-Formeln.
WT: Gut, Herr Holtmann. Ich denke, damit sind wir dann durch. Warten Sie doch bitte eine Minute draußen...

So in etwa ist meine Prüfung abgelaufen. An alle Details konnte ich mich nicht mehr erinnern, insbesondere nicht an die Reihenfolge. Insbesondere in Angewandte Automatentheorie hat er mich sehr viele Sachen gefragt, die über den Vorlesungsstoff hinausgingen. Die Prüfungsatmosphäre war sehr angenehm. Professor Thomas gibt gerne Hilfestellungen, wenn man nicht weiter weiß. Bei ihm ist es wichtig, die Zusammenhänge zwischen den einzelnen Themen zu kennen, Details werden nur sehr selten verlangt. Man sollte sich aber immer möglichst präzise ausdrücken. Bis auf ein paar Schwächen bei Angewandte Automatentheorie bei den über den Vorlesungsstoff hinausgehenden Fragen, ist die Prüfung recht glatt verlaufen. Professor Thomas fand es insbesondere gut, dass ich mich auch mit einigen weiterführenden Fragestellungen der verschiedenen Themengebiete beschäftigt hatte, obwohl sie nicht so direkt im Prüfungsumfang enthalten waren.

Verwendete Materialien:

1. Automata and Reactive Systems
   Nach Absprache mit Professor Thomas war hier die gesamte Vorlesung aus dem Wintersemester 2002/2003 bis auf Teil IV des Folien-Skriptes prüfungsrelevant. Wenn auch nicht so detailliert, hatte ich aber auch diesen Teil vorbereitet. Die Vorlesung habe ich im Wintersemester 2002/2003 gehört und anhand der Vorlesungsvideos, den Folien, den Übungsaufgaben und den Prüfungsprotokollen ein kommentiertes Skript erstellt. Ich habe dann mit dem kompletten, kommentierten Skript gelernt (also ohne Zusammenfassungen), um auch die Details draufzukriegen. Prüfungsgrundlage war eigentlich das offizielle Skript, aber die Folien sind meist noch etwas ausführlicher.

2. Angewandte Automatentheorie
   Die Vorlesung habe ich teilweise im Sommersemester 2003 gehört und im Sommersemester 2005 mit den beiden Vorlesungen “Advanced Theory of Finite Automata“ und “Unendliche Transitionssysteme“ wiederholt. Auch hier bin ich lernmäßig wie in Automata and Reactive Systems vorgegangen. Prüfungsgrundlage war wieder das offizielle Skript, aber auch hier sind die Folien etwas ausführlicher.

3. Model Checking
   Die Vorlesung habe ich nie gehört. Prüfungsrelevant war die komplette Vorlesung bis einschließlich Kapitel 7.3, die ich mir anhand der “Lecture Notes“ von der Vorlesungsseite erarbeitet habe. Auch hier habe ich mir ein kommentiertes Skript geschrieben. Die Übungen habe ich aus Zeitgründen aber nicht selber gerechnet, sondern mir nur die Musterlösungen angeschaut.

Ich hoffe, das Protokoll hilft Euch weiter. Viel Erfolg!!!!