# Gedächtnisprotokoll
## Vertiefungsprüfung

**Prüfer**: Professor Thomas
**Datum**: 30.10.2006
**Vorlesungen**: Applied Automata Theory (Advanced Theory of Finite Automata & Unendliche Transitionssysteme), Automata and Reactive Systems (Automata on Infnite Words & Unendliche Spiele), Model Checking (Prof. Katoen), außerdem sollte ich mir mal die Folien zur Model Checking Vorlesung von Professor Thomas angucken
**Dauer**: 40 Minuten
**Note**: 1,0

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

**Professor Thomas**: Fangen wir mit der Angewandten Automatentheorie an. Was fällt Ihnen zur Minimierung von NFA's ein?
**Ich**: Ein schwieriges Problem, es ist PSPACE-vollständig.
**Er**: Ja, dazu kommen wir später. Wie kann man es dennoch angehen.
**Ich**: Zum einen könnte man eine Bisimulation für den Automaten berechnen und die bisimilaren Zustände verschmelzen.
**Er**: Wie ist da die Komplexität?
**Ich**: Das geht in Polynomialzeit.
**Er**: Gibt es noch eine andere Möglichkeit?
**Ich**: Man könnte den universellen NFA berechnen.
**Er**: Wie geht denn das?
**Ich**: Man dreht den NFA, determinisiert ihn, macht das noch mal und kann dann darin irgendwie den minimalen NFA finden.
**Er**: Den minimalen? Ist der eindeutig?
**Ich**: Nein, stimmt. Es kann mehrere minimale NFA's geben?
**Er**: Für welche Sprache zum Beispiel?
**Ich**: $a^n(a^*)$. Da müsste es $n$ minimale NFA's für geben. Eine andere Möglichkeit wäre, alle kleineren NFA's durchzugehen und prüfen, ob sie ein minimaler NFA für die Sprache sind, dazu muss man aber jedes mal einen Äquivalenztest machen, der genau so schwer wie das Minimierungsproblem ist.
**Er**: Kommen wir mal zur Komplexität. Wie zeigt man das?
**Ich**: Man zeigt, dass das Nicht-Universalitätsproblem für NFA's ein Spezialfall des Minimierungsproblem ist und reduziert deswegen das Wortproblem für eine beliebige Sprache aus PSPACE auf das Nicht-Universalitätsproblem. Habe dann kurz aufgeschrieben, was man zeigen muss: $w \in L \Leftrightarrow F(w)$ nicht universal (musste aber nicht zeigen, wie es geht).
**Er**: Gehen wir mal zu Systemen mit unendlichen vielen Konfigurationen. Welche haben wir denn da kennengelernt, bei denen Erreichbarkeit entscheidbar

ist?
**Ich**: Pushdownautomaten beziehungsweise Pushdownsysteme, wenn wir nicht an Sprachakzeptoren interessiert sind.
**Er**: Und wie zeigt man dann, dass die Erreichbarkeit entscheidbar ist?
**Ich**: Ein Satz von Büchi sagt: Wenn $C$ eine reguläre Menge von Konfigurationen ist, dann ist auch $pre^*(C)$ und $post^*(C)$ wieder regulär. Außerdem kann man aus einem Automaten für $C$ Automaten für $pre^*(C)$ und $post^*(C)$ konstruieren, indem man den Automaten saturiert.
**Er**: Zeigen sie das mal für $post^*(C)$.
**Ich**: (Skizze zu gemalt und erklärt, welche Transitionen man hinzufügen muss, wenn der P-Automat die Transition $(p, a, r)$ hat und es die Regel $pa \to qw$ im PDS gibt)
**Er**: Kann man auch entscheiden, ob man von einer Konfiguration aus unendlich viele Konfigurationen erreichen kann?
**Ich**: Ja, dazu muss man $post^*$ der Konfiguration berechnen und dann eine Schleife im Automaten suchen.
**Er**: Wie ist denn Akzeptanz bei Pushdownautomaten definiert?
**Ich**: Durch erreichen des leeren Kellers oder äquivalent durch erreichen eines Endzustandes.
**Er**: Was wäre denn, wenn wir eine reguläre Menge von Anfangskonfigurationen und eine reguläre Menge von Endkonfigurationen haben und damit Akzeptanz definieren?
**Ich**: Zuerst versucht, damit eine kontextsensitive Sprache zu erkennen.
**Er**: Was müsste man denn zeigen?
**Ich**: Entweder zeigen, dass man eine nicht-kontextfreie Sprache so erkennen kann oder zeigen, dass man diese Akzeptanzbedingung auf eine andere reduzieren kann.
**Er**: Was denken Sie?
**Ich**: Dass man damit nicht mehr erkennen kann.
**Er**: Ja, stimmt. Beim Pushdownsystemen ersetzt man ja Präfixe, wie sieht es aus, wenn man Infixe ersetzt?
**Ich**: Damit könnte man Turingmaschinen simulieren, also unentscheidbar.
**Er**: Ok, und wenn man Präfixe und Suffixe ersetzen darf?
**Ich**: Dann kann man eine Queue simulieren und damit auch eine Turingmaschine.
**Er**: Und wenn wir keine globale Kontrolle (streicht den Zustand $q$ in der Skizze durch) haben?
**Ich**: Dann könnte man den Zustanden im ersten Buchstaben des Wortes mitkodieren.
**Er**: Ok, aber man kann ja auch nur am Ende was ersetzen.
**Ich**: Ok, damit klappt die Kodierung dann nicht.
**Er**: Ja, man kann zeigen, dass dann Erreichbarkeit entscheidbar ist, indem man einen Saturierungsalgorithmus benutzt. Da arbeitet Herr Altenbernd dran und Büchi hat es früher wohl geahnt, dass es geht. Wir hatten auch Automaten mit unentscheidbarem Erreichbarkeitsproblemen.
**Ich**: Ja, kommunizierende Automaten zum Beispiel. Da kann man schon mit einfachen Automaten Turingmaschinen simulieren.

**Er**: Ja, dann kommen wir zu Petrinetzen. Da hatten wir in einem Schwierigen Beweis das Lemma von Dixon. Was sagt es und wofür haben wir es benutzt?
**Ich**: Es sagt, dass $\leq$ auf $\mathbb{N}^r$ eine Wohlquasiordnung ist. Damit haben wir gezeigt, dass der Karp-Miller-Baum endlich ist.
**Er**: Kommen wir zu Automaten auf unendlichen Wörtern. Wann akzeptiert ein Co-Büchi-Automat?
**Ich**: Er akzeptiert $\alpha$, wenn der Lauf ab einem bestimmten Zeitpunkt nur noch Endzustände besucht.
**Er**: Dann geben Sie mal zwei Sprachen an, die von einem Co-Büchi-Automaten erkannt beziehungsweise nicht erkannt werden.
**Ich**: $(0+1)^*0^\omega$ und $(0^*1)^\omega$.
**Er**: Wie zeigt man denn, dass die zweite nicht erkennbar ist?
**Ich**: (Kurz überlegt) Wir haben es in der Übung mit einem Abzählargument gezeigt.
**Er**: Ja, wie kann man denn entscheiden, ob eine Sprache Co-Büchi-erkennbar ist?
**Ich**: Mit dem Satz von Landweber. Mit dem kann man entscheiden, ob eine Sprache A-, E-, Büchi- oder Co-Büchi-erkennbar ist.
**Er**: Dann formulieren sie den Satz mal bitte für Co-Büchi.
**Ich**: $\mathcal{F}$ abgeschlossen unter Subloops genau dann wenn $L$ Co-Büchi-erkennbar ist.
**Er**: Wie beweisen Sie die einfache Richtung?
**Ich**: Einfach ist von Co-Büchi-erkennbar nach $\mathcal{F}$ abgeschlossen unter Subloops (hab dann erst mal bewiesen, dass der Muller-Automat, den man direkt aus dem Co-Büchi-Automaten konstruiert, unter Subloops abgeschlossen ist, aber nicht die allgemeine Aussage des Satzes, dass es für alle Automaten gilt)
**Er**: Machen wir mit den Spielen weiter. Normalerweise hat man einen deterministischen Automaten mitlaufen, der entscheidet, ob eine Partie von Spieler 0 gewonnen wird. Kann man das ganze auch mit nichtdeterministischen Büchi-Automaten aufziehen und Strategien definieren?
**Ich**: Nein, da nicht eindeutig ist, in welchem Zustand sich der Automaten nach einem Präfix befindet.
**Er**: Wie würden sie vorgehen, wenn Sie einen nicht-deterministischen Büchi-Automaten hätten?
**Ich**: Ich würde den determinisieren und einen Muller-Automaten und damit auch ein Muller-Spiel erhalten.
**Er**: Wie groß ist der?
**Ich**: $2^{\mathcal{O}(n \log n)}$ wenn der Büchi-Automat $n$ Zustände hatte.
**Er**: Und wie lösen sie das Spiel dann?
**Ich**: Durch eine Reduktion auf ein Paritätsspiel.
**Er**: Wie sieht dann die Gesamtkomplexität aus?
**Ich**: $(2^{\mathcal{O}(n \log n)})!$.
**Er**: Dann haben wir Algorithmen für Paritätsspiele kennengelernt. Können sie dazu was sagen?
**Ich**: Ja, es gibt den naiven Ansatz eine Strategie zu raten und dann zu verifizieren, ob sie eine Gewinnstrategie ist.
**Er**: Was zeigt dass über die Komplexität.

**Ich**: Das das Problem in NP$\cap$ co-NP ist, es ist aber sogar in UP$\cap$co-UP.
**Er**: Wofür steht denn das $U$?
**Ich**: Für unambiguous, das heißt es gibt eine nichtdeterministische polynomialzeitbeschränkte Turingmaschine, die höchsten eine akzeptierende Berechnung hat. Äquivalent dazu ist, dass es einen eindeutigen kurzen Zeugen gibt.
**Er**: Wie zeigt man, dass es einen solche Strategie (den Zeugen) gibt?
**Ich**: Durch eine Reduktion auf Mean-Payoff-Games, von denen man weiß, dass sie in UP$\cap$co-UP sind. (Wir haben dann zusammen überlegt, wie man diese Strategie direkt auf dem Spielgraphen charakterisieren kann)
**Er**: Wofür haben wir denn die Paritätsspiele benutzt?
**Ich**: Um zu zeigen, dass Paritätsbaumautomaten unter Komplement abgeschlossen sind.
**Er**: Damit haben wir dann gezeigt, dass eine Theorie entscheidbar ist.
**Ich**: Ja, die MSO-Theorie des binären Baumes ist entscheidbar.
**Er**: Und wie sieht es mit diesen beiden Strukturen aus (malt den unendlich verzweigenden Baum S$\omega$S und den Binärbaum mit Equal-Level-Prädikat auf)?
**Ich**: Den ersten kann man in den Binärbaum einbetten, er hat also eine entscheidbare Theorie, beim zweiten dürfte es schwierig werden, da die Baumautomaten nicht beliebig zeit nach unten zählen können.
**Er**: Welche Struktur mit unentscheidbarer Theorie kennen Sie denn?
**Ich**: Das unendliche Grid. Das könnte man wohl auch in dem zweiten Baum einbetten, also hat der auch eine unentscheidbare Theorie.
**Er**: Ja, richtig. Kommen wir noch zum Model-Checking. Wie geht CTL-Model-Checking?
**Ich**: Man berechnet induktiv die $Sat$-Mengen der Teilformeln. Es gilt $Sat(true) = S$, $a$ gilt an allen Knoten, die mit $a$ beschriftet sind, beim $\wedge$ berechnet man den Schnitt der beiden $Sat$-Mengen...
**Er**: Machen Sie mal die interessanten Fälle.
**Ich**: Da man sich auf existentielle Formeln beschränken kann, haben wir da $\mathbf{X}\varphi$, da bestimmen wir alle Knoten, die einen Nachfolger haben, an dem $\varphi$ gilt. Bei $\mathbf{E}(\varphi\mathbf{U}\psi)$ nehmen wir $Sat(\psi)$ und vergrößern dass schrittweise, indem wir die Knoten hinzufügen, an denen $\varphi$ gilt (mit einer Skizze erklärt).
**Er**: Ok, und wie ist die Komplexität?
**Ich**: Wir müssen jede Teilformel einmal betrachten, also schon mal linear in der Formellänge, und die Bestimmung der $Sat$-Mengen geht in linearer Zeit in Bezug auf die Größe des Transitionssystems, also insgesamt linear in der Länge der Formel und in der Größe des Transitionssystems.
**Er**: Und wie sieht es mit LTL aus?
**Ich**: Da baut man aus der Formel einen generalisierten Büchi-Automaten und aus diesem einen Büchi-Automaten.
**Er**: Und wie groß ist der?
**Ich**: Der hat $2^m$ Zustände, wenn die Formel $m$ Teilformeln hat. Man rät damit die $\varphi$-Expansion des Eingabewortes.
**Er**: Ja, gibt es noch einen anderen Weg?
**Ich**: Ja, man kann aus der Formel einen alternierenden Automaten konstruieren und wegen der speziellen Form dieser Automaten mit Potenzmengenkonstruktion einen Büchi-Automaten. Asymptotisch sind die also gleich groß.

**Er**: Ja, wie sieht denn die Komplexität dabei aus?
**Ich**: PSPACE-vollständig.
**Er**: Wie zeigt man das?
**Ich**: Man reduziert wieder das Wortproblem für eine Sprache aus PSPACE auf das Model-Checking-Problem (hab das Transitionssystem mit den „Diamanten" aufgezeichnet und kurz erläutert, was man mit den Formeln ausdrücken muss).
**Er**: Gut, dann hatten wir noch die Sicherheits- und Lebendigkeitseigenschaften. Man kann jede Eigenschaft als eine Kombination solcher darstellen.
**Ich**: Ja, es gilt $P = P_{safe} \cap P_{life}$.
**Er**: Wie ist eine Lebendigkeitseigenschaft definiert.
**Ich**: Sie schließt keinen endlichen Präfix aus, also $Pref(P) = \Sigma^*$.
**Er**: Ok, dann gehen Sie mal bitte raus.

Die Prüfung war sehr angenehm und hat Spaß gemacht: Es kamen viele Sachen dran, die in den Vorlesungen nicht vorkamen, außerdem musste ich selten etwas richtig formal aufschreiben oder zeigen. Professor Thomas hat viel mehr gefragt, ob etwas geht oder wie man es zeigen kann und wollte wissen, ob ich die Zusammenhänge verstanden habe und auch weitergehende Sachen verstehe.