Protokoll Theorieprüfung Angewandte Automatentheorie, Model Checking, Automata on Infinite Objects, Termersetzungssysteme

Prüfer: Prof. Thomas/Giesl

Beisitzer: keine

Note: 1.0

Die Prüfungssituation war sehr entspannt. Insbesondere mit Professor Thomas machte die Prüfung richtig Spaß, da er bei mir wenig in's Detail ging und eher über Theorien auf hohem Niveau mit mir diskutierte. Das war echt interessant, wenngleich ich aber auch manchmal aufgrund der großen Sprünge im Stoff etwas Probleme hatte, ihm zu folgen. Bei Prof. Giesl hab ich dann leider zwei, drei Mal etwas Quatsch erzählt, was wohl an der Nervosität lag. Aber er hat mich dann immer ganz schnell davon überzeugt, dass ich das doch nochmal überdenken sollte und da ich dann jedesmal beweisen konnte, dass das Quatsch war und es dann auch richtigstellen konnte, war er wohl trotzdem mit dem Ergebnis zufrieden, was sich ja dann auch in der mehr als fairen Note wiederspiegelt. Insgesamt alles eine sehr positive Erfahrung.

Prüfungsteil Prof. Thomas

Wir haben ja in der Vorlesung verschiedene Beschreibungslogiken kennengelernt. Welche davon beschreibt denn im Allgemeinen die regulären Sprachen?

Das ist die MSO, denn die MSO-Formeln über endlichen Wörtern sind genauso ausdrucksstark wie DFAs/NFAs.

Achso? Ist es denn dabei egal, ob ich einen NFA oder DFA nehme?

Naja, also so wie wir das Problem angegangen sind, mit der Partition Formel, ist das nicht egal, denn wenn wir die Partitionierung voraussetzen, wirkt sich das ja auch auf die Restformel aus.

Partition?

(Ich Formelanfang hingeschrieben: $\exists Y_1 \exists Y_2 \exists Y_3 ...$)

Achso, ja. Jetzt sehe ich, was sie meinen. Aber wofür stehen denn hier die Existenzquantoren?

Ja jeder Quantor beschreibt einen Zustand im DFA. Die Partition-Formel drückt dabei die Eindeutigkeit des Laufes aus.

Ja, das stimmt, aber wie könnte man es denn noch ausdrücken?

Hhmmm, ich weiß nicht, worauf Sie hinaus wollen.

Naja, man könnte auch sagen 'Es existiert ein Lauf!'

Achso! Ja ok...

Ja und ist die Konstruktion nun auch mir einem NFA möglich?

Ja, also ich wüsste nicht, was dem im Wege stehen würde. (Hinterher wurde mir erst klar, dass er mich mit dem 'Es existiert ein Lauf' dorthin pushen wollte, denn das ist ja genau die Akzeptanzbedingung für einen NFA.)

Ja genau. Wie würde man denn nun so einen NFA wieder in eine Formel umwandeln?

(Ich erklärt: Rückführung auf MSO_0, verschiedenen Fälle kurz angerissen...)

Ja und wie nennt man das jetzt? Wie geht das jetzt?

(Hier hab ich ne ganze Weile nicht verstanden, was er eigentlich hören will, bis ich dann endlich die erlösenden Worte sagte: 'Konstruktion induktiv über den Formelaufbau' - Das war's wohl.)
Wie ist die Komplexität der Umwandlung den DFA?
Nicht elementar. Das liegt daran, dass ich bei der Projektion einen NFA bekomme...
... dann mach ich halt mit nem NFA weiter...
..., den ich dann bei einer Negation wieder determinisieren muß, was exponentiell aufwendig ist. Bei Allquantoren wird's dann fies, denn $\exists x\phi == \neg\forall x\neg\phi$...
Achso, ja. Ok. Nun hatten wir ja in Model Checking auch noch andere Logiken. Welche waren denn das?
CTL, LTL, CTL*, TCTL
Ja, dann schreiben Sie doch mal eine Formel für 'Endlich oft p'!
FGp
Ja. Und 'es existiert auf einem Pfad ein p' in CTL?
EFp
Und 'auf einem Pfad immer wieder p'? EGFp - das ist allerdings nicht mehr CTL, sondern CTL*. In CTL ist das nicht ausdrückbar. Das geht nur mit nem Allquantor vorne.
Ja, wie hängen CTL und LTL uind CTL denn genau zusammen?*
(Ich grob erklärt: CTL, LTL schief zueinander, CTL* umfasst beide; wenn eine CTL-Formel zu LTL-Formel äquivalent, dann reicht der Äquivalenztest zur Formel, die entsteht durch Weglassen der Pfadquantoren...)
Ja wenn Sie denn nun zwei solche LTL-Formeln haben. Wie genau würden Sie denn nun die Äquivalenz testen?
(Eine Weile überlegt, denn das hatten wir so nicht...) Ja, man könnte ja die entsprechenden Büchi-Automaten generieren und die dann auf Äquivalenz testen.
Ja, das könnte man wohl. Ok, aber wir waren ja eigentlich noch gar nicht bei Model Checking... Zurück zu den regulären Sprachen. Ich meine, wir hätten da eine in sich abgeschlossene Unterklasse kennengelernt. Welche Logik definiert die denn?
FO[',<] bzw. LTL, denn mit denen kann man Counting nicht ausdrücken.
Ja. Und wie würden Sie nun algorithmisch feststellen, ob eine durch eine MSO-Formel beschriebene Sprache auch FO-definierbar ist?
Naja, also dann wenn Sie keine Setquantifier benutzt, ist sie ja schon FO.
Ja, aber es kann ja sein, dass sie die zwar benutzt aber eigentlich nicht braucht.
Naja, dann könnte man den zugehörigen DFA generieren und schauen, ob der Transition Monoid nur triviale Gruppen enthält. Falls das der Fall ist, dann ist das FO definierbar.
Ja, so würde man das wohl machen. Wie kann man diese Klasse noch definieren?
star-free, permutation-freier DFA, ...
Und wenn ich in der FO-Logik das < nun weglasse?
Na dann hat man ein Problem, denn ohne einen Setquantifier kann man < nicht mit dem Successor simulieren.
Ja, richtig. Welche Sprache könnte man denn so z.B. nicht ausdrücken?

Na z.B. 'Alle Wörter über {a,b}, in denen mehr b vorkommen als a'.
(Lacht) Vorsicht.
Achso! Ja klar, das ist nicht mehr regulär. Sie wollen sicher was reguläres... Hmmm...
Ja, man kann dann z.B. sowas nehmen wie 'Die Sprache über {a,b,c}, in der vor einem b ein a steht (mit evtl. auch c's dazwischen).'
Ja - klar.
Gut, dann kommen wir mal zu Sprachen über unendlichen Bäumen. Ist denn da die Äquivalenz entscheidbar?
(Intuitiv war mir das klar, aber wie... Nach einiger Zeit dachte ich dann...) Naja, man könnte zeigen, dass alle Paritätsspiele, die Player Automaton auf dem Paritätsautomaten zur einen Sprache gewinnt auch auf dem der anderen Sprache gewinnt.
(Lacht) Ja, das ginge wohl. Aber wie könnte man das denn noch machen?
Naja, das Problem ist ja, dass der MTA zu der Sprache ja nichtdeterministisch ist, deswegen bringt einen eine State-Equivalence da ja wohl nicht weiter...
Achso, Sie versuchen jetzt, da was effizientes zu finden. NDefinitionein sowas geht wohl nicht. Aber wenn $T_1 = T_2$, dann ja auch $T_1 \subseteq T_2$ und umgekehrt...
(Nun sind wir dann 'zusammen' drauf gekommen, dass man ja über Leerheitstest des Schnitts von T_1 mit dem Komplement von T_2 diese Inklusion beweisen kann. Da die Klasse entsprechend abgeschlossen ist, geht das.)
Gut, dann von unendlichen Bäumen mal zu unendlichen Automaten. Wir hatten ja diese CFSMs oder wie auch immer die heißen. Ist denn da der Leerheitstest entscheidbar?
Nein. (Kurz die Beweisidee geschildert)
Ja, richtig. Jetzt besteht ja diese CFSM, wie Sie gesagt haben aus einer Queue und einer solchen Logik mit Kontrollzuständen. Wenn man denn jetzt die Logik wegließe, wie säh denn dann das Erreichbarkeitsproblem aus? Entscheidbar?
(Kurz überlegt...) Ja, das ist ja so in etwa wie bei den Lossy Queue, über die wir uns mal kurz unterhalten hatten. Wenn ich die Logik weglasse, dann wird der Statespace ja irgendwie... flacher... durchgängiger... einfacher zu beschreiben. Dann sollte das eigentlich wieder gehen. Denke ich mal so intuitiv.
Ja, das ist interessant, denn ich wollte das in der nächsten Vorlesung mal machen und ich fand das gar nicht intuitiv. Aber schön! Wenn ich denn jetzt aus dieser Queue zwei Stacks mache, was passiert denn dann?
(Kurz Reduktion erklärt, bis runter auf den 2-Counter. $->$ Unentscheidbar)
Das ist ja interessant. Wie sieht das denn aus - wir haben doch auch Petrinetze kennengelernt. Das sind doch irgendwie auch Counter. Bei n Stellen sogar n-Counter. Wieso ist denn da das Erreichbarkeitsproblem entscheidbar?
Naja, der Unterschied liegt hier beim Test auf Null. Während ein Petri-Netz bei Null Tokens im Pre-Set einfach die Transition nicht schaltet und wartet, hat eine CFSM den Test 'If X=0 then foo ELSE bar', man kann also verzweigen. Das macht das Modell reicher.
Na gut. Dann nochmal zum Model Checking. Mit den AFAs für das LTL-MC, wie geht das denn nun so genau?
(Ich erklärt, wie das umgewandelt wird...)

Und wie ist das mit den Loops?
Naja, man hat zwei Typen von Loops, einmal die Final Loops für Release Subformeln und dann die aus Q_{Loop} für die Until-Formeln. Alle diese Loops sind elementar, so dass der AFA ein partiell geordneter AFA wird. (Dann noch kurz was zur Komplexität erzählt und zu den weiteren Schritten - aber sehr kurz.)
Und bei den Timed Automata... wie erhält man da denn den endlichen Statespace?
Man hat ja stets gegeben eine Menge von Uhrenbedingungen - durch die Spec und durch die Anfrage. Dies gibt einem ein k_x für jede Uhr x, so dass man dann den Space schonmal nach oben beschränken kann. Innerhalb dieser Region kann man sich dann auf 'x hat Wert n' und 'x ist zwischen Werten n und m' beschränken. Dabei muß man noch darauf achten, dass die Reihenfolge der einzelnen Uhren eines Uhrenstandes gleich bleibt.
(Irgendwo zwischendurch kam auch noch was zu AFAs: Akzeptanzbedingung und warum man die AFAs trotzdem nimmt, obwohl die doch eine so 'fürchterlich komplizierte' Akzeptanzbedingung haben...)

Prüfungsteil Prof. Giesl (TES)

Schreiben Sie doch mal ein Gleichungssystem auf für Minus. Dann noch eine Anfrage, ob x minus x gleich 0 ist.
(Ich hingeschrieben.)
Ok - folgt denn die Gleichung nun aus dem System? Warum nicht? Haben Sie Minus falsch definiert?
Nein, das liegt daran, dass manche Algebren, die das System erfüllen, die Anfrage nicht erfüllen. Minus muß ja nicht wirklich minus bedeuten.
Wie würde ich nun zeigen, dass das nicht gilt?
(Alg. Wortproblem; Satz von Manna und Ness, Terminierung; Lokale Konfluenz / Kritische Paare, war hier alles sehr lustig, weil es keine CP gab und man die Terminierung schon mit Embedding zeigen konnte.)
Ja und wie sieht das denn aus, wenn ich keine Terminierung habe?
Naja, dann reicht auch die Linkslinearität. Wenn ich dann lokale Konfluenz habe, dann geht das auch alles. (Anm.: Das stimmt so nicht, kam dann auch im Folgenden raus...)
Ach wirklich? Dann malen Sie doch mal ein System auf, das linkslinear und lokal konfluent aber nicht konfluent. Kennen Sie eins? (Ich abcd-Beispiel aufgemalt.)
Also?
Ja, klar, das stimmte so nicht. Denn offensichtlich ist das System ja linkslinear und auch lokal konfluent aber nicht konfluent...
Also kann man Konfluenz nun immer entscheiden?
Nein.
Ist es denn semi-entscheidbar? Kann man denn immer kritische Paare bilden?
Ja, das geht immer. Und dadurch ist das auch semi-entscheidbar, denn wenn ein CP zusammenführbar ist, dann bekomme ich das durch Reduktion auch immer hin.
Ja genau. Wie ist das denn - kann ich denn zu einem Terminierenden TES

immer eine Reduktionsordnung (man beachte: Ordnung) finden, so dass das System mit dieser wie im Satz von Manna und Ness orientierbar ist?
Ja, das geht, weil im Falle, dass R terminiert schon $\rightarrow_R$ eine entsprechende Reduktionsordnung ist.
Ach wirklich? Welche Eigenschaften hat denn eine Reduktionsordnung?
Stabil, monoton, fundiert und transitiv
Wieso ist $\rightarrow_R$ stabil / monoton?
Das liegt am Satz von Birkhoff. (Anm: das war Quatsch)
Moment, was sagt denn der aus?
(kurz erklärt)
Also sagt der doch über TESe gar nichts aus. Also woran liegt das nun?
Ja, stimmt. Es liegt an der Definition von $\rightarrow_R$. Die impliziert das ja gerade.
Ja genau. Und ist $\rightarrow_R$ denn nun fundiert/transitiv?
Naja, die Fundiertheit folgt ja sofort aus der Terminierung von R und die Transitivität... Achso, ja. Da muß man natürlich noch die Transitive Hülle $\rightarrow_{R^+}$ betrachten...
Das war's worauf ich hinaus wollte. Wenn Sie die nun bilden, ist die dann auch noch fundiert? Ja? Ist das immer so?
Nein, im allgemeinen, glaube ich, nicht.
Moment, also bei der Ersetzungsrelation über R ist das so aber allgemein nicht? Das ist aber interessant...
(Ich dann kurz überlegt und dann bewiesen, dass das doch immer der Fall ist, denn wenn in der Transitiven Hülle eine unendliche Kette existiert, dann gab's die auch vorher schon.)
Also gilt die Gleichung nun?
Ja.
Wieso das? Eben meinten Sie doch noch, sie gelte nicht. Was haben Sie denn hier mit was ersetzt?
(Hier war ich gerade unheimlich nervös, hatte unifiziert, wo man nur matchen durfte, daher der dumme Fehler.)
Ja, sie dürfen das hier nicht mit der Logikprogrammierung verwechseln. Da unifiziert man. Das macht man hier nicht.
Ja, da hatte ich mich wohl vertan. Aber immerhin gilt die Gleichung induktiv.
Ja erklären Sie doch mal, wie das funktioniert.
(Ich kurz erklärt: Rückführung auf Algebren über Grundtermen, Definitionsprinzip.)
Warum hat man denn dieses seltsame Definitionspronzip?
Naja, damit stellt man halt sicher, dass jede Reduktion auf einem Term irgendwann mal zu einer Grundidentität führt, die nur Konstruktorgrundterme enthält. Diese kann man dann eindeutig auf Inkonsistenzen prüfen.
Jetzt hatten wir ja dazu diese Reduktionsregeln. Kann man die in beliebiger Reihenfolge anwenden?
Ja. (Kurz noch was zu Fairness erzählt.)
Welche Fälle gibt es nun also, wie sich der Algorithmus für die implizite Induktion verhalten kann?
Entweder Nichtterminierung $->$ unendliches, äquivalentes TES (Äquivalenz

folgt aus beagter Fairness), oder Terminierung mit Success (endliches, äq. TES) oder FAIL weil die Reduktionsordnung evtl. zu schwach war $->$ keine Aussage.
Oder?
Achso, oder natürlich das explizite Fail wegen einer Inkonsistenz.
Bei einem solchen Fail, gilt denn dann die Gleichung in jedem Fall nicht?
Ja, das ist dann sicher.
Ja, richtig.

Viel Erfolg!
Rückfragen an: eric@bodden.de