Gedächnisprotokoll
Termersetzung, Funktionale, Compilerbau
Prüfer: Prof.Giesl
Prüfling: Markus de Brün
Note: 1,0
Dauer: 45 min

Das Protokoll ist vermutlich nicht vollständig, aber das wichtigste sollte drin sein.
Die Prüfung war insgesamt sehr angenehm. Ich hatte ausreichend Zeit mir meine Antworten zu überlegen, habe auch nicht alles gewusst und musste mir einige Sachen in der Prüfung noch erarbeiten. Das war aber wohl nicht so schlimm. Ich kann eine Prüfung bei Prof.Giesl sehr empfehlen und wünsche allen viel Erfolg. Fragen oder Korrekturvorschlägen bitte an markus.bruen@rwth-aachen.de

Giesl: Womit wollen sie denn anfangen?

Ich: TES

Gut. Dann machen wir TES, FP und dann CB.
Worum ging es denn in TES?

Wir wollten das Wortproblem lösen.

Das heißt?

Automatisch zu überprüfen, ob aus einer Gleichungsmenge eine Gleichung folgt.

Und was heißt *folgt*?

Das ist semantisch definiert. Eine Gleichung folgt aus einer Gleichungsmenge ($\mathcal{E}$), wenn alle Algebren, die Modell von $\mathcal{E}$ sind auch die Gleichung erfüllen.

Genau. Ist das denn entscheidbar?

Nein. Das ist im Allgemeinen unentscheidbar.

Ja. Aber das ist ja nicht schön. Ist das vielleicht ein bisschen besser, als unentscheidbar?

Es ist semi-entscheidbar, wenn wir nur Gleichungen zulassen, die links und rechts die gleichen Variablen haben. Wir können dann einen Suchbaum aufbauen.

Wie sieht das denn mit dem Verzweigungsgrad aus? Was heißt eigentlich *semi-entscheidbar*?

Semi-entscheidbar heißt, dass wir die Lösung finden, wenn die Aussage wahr ist. Wenn wir $s \equiv_{\mathcal{E}} t$ untersuchen wollen starten wir mit s und wenden dann Gleichungen darauf an. Das geht, weil wir nur einen endlichen Verzweigungsgrad haben da nur endlich viele Gleichungen auf einen Term anwendbar sind. Wenn wir dann irgendwann den Term t finden folgt die Gleichung und sonst suchen wir halt weiter.

Aber die Einschränkung zu dem Variablen, die sie gemacht haben... Die gefällt mir gar nicht. Die ist wirklich streng! Schreiben sie mal ein Gleichungssystem für Subtraktion auf.

$m(x, 0) \equiv x$
$m(0, y) \equiv 0$
$m(s(x), s(y)) \equiv m(x, y)$

Okay. Wie sieht das denn hier aus?

Also wenn wir es schaffen ein äquivalentes, konvergentes TES zu konstruieren, dann ist das sogar entscheidbar.

Ja, aber das machen wir später. Bleiben wir mal bei Gleichungen. Gilt ihre Bedingung denn hier?

Nein. In der zweiten Gleichung gilt die nicht.

Und? Ist das jetzt unentscheidbar? Wie sieht das denn jetzt mit dem Verzweigungsgrad aus?

Der Grad ist unendlich, weil sich 0 mit der zweiten Gleichung zu beliebig vielen Termen auswerten lässt. Trotzdem ist das immer noch semi-entscheidbar, weil wir den Suchbaum geschickt aufbauen können.

Diagonalisierung, genau. Warum machen wir denn jetzt was anderes - sie haben da TESe erwähnt...

Weil wir gerne ein Entscheidungsverfahren für das Wortproblem hätten. Dafür brauchen wir halt ein äquivalentes, konvergentes TES.

Und was müssen wir jetzt machen?

Erstmal die Gleichungen richten und dabei aufpassen, dass das ganze terminiert. Dazu suchen wir halt eine Reduktionsordnung, welche die Ersetzungsrelation enthält.

Machen sie mal...

$m(x, 0) \to x$
$m(0, y) \to 0$
$m(s(x), s(y)) \to m(x, y)$

Hätten sie hier irgendwas anders richten können?

Nein. Das würde in allen drei Fällen die Terminierung zerstören.

Okay. Wann terminiert so ein TES denn?

Eben wenn wir eine Reduktionsordnung $\succ$ finden können, so dass für alle Regeln $l \to r \in \mathcal{R}$ gilt $l \succ r$. Wir suchen also nach Reduktionsrelationen, die die Ersetzungsrelation enthalten.

Sie haben jetzt einmal Reduktionsrelation und einmal Reduktionsordnung gesagt. Was denn nun?

Ist egal. Stimmt beides.

Ja. Dann nehmen wir mal eine Reduktionsordnung. Wieso terminiert ein TES jetzt?

Satz von Manna und Ness. Nur mit Ordnungen statt Relationen.

Zeigen sie mal die Hinrichtung.

Also wenn das TES terminiert, dann wählen wir $\succ$ gleich der transitiven Hülle der Ersetzungsrelation, also als $\to_{\mathcal{R}}^{+}$.

Welche Eigenschaften muss die haben?

Stabil, monoton, transitiv, fundiert. Fundiert ist sie, weil...

Ja. Das reicht schon. Gilt der Satz denn auch, wenn wir statt Reduktionsordnungen Simplifikationsordnungen nehmen?

Nein. Das gilt nicht, weil es TESe gibt, die terminieren ohne simpel zu terminieren.

Haben sie ein Beispiel?

$f(f(x)) \to f(g(f(x)))$.

Warum denn nicht?

Weil auch $f(g(f(x))) \succ_{emb} f(f(x))$ gelten würde.

Gut. Also weiter. Was brauchen wir denn in unserem Beispiel noch?

Konfluenz. Dazu müssen wir kritische Paare bilden.

Nur zu...

Also die ersten beiden Gleichungen überlagern. Da kommt dann dieses kritische Paar raus. (Weiß nicht mehr, was ich geschrieben habe, aber es war falsch.)

Wirklich? Das sehe ich nicht. Was ist denn ein kritisches Paar?

Also wenn es Regeln $l_1 \to r_1$ und $l_2 \to r_2$ gibt, so dass $l_1|_\pi\sigma = l_2\sigma$ gilt, dann ist das kritisch Paar $< r_1\sigma, l_1[r_2]_\pi\sigma >$.

Richtig. Wie wäre dass hier?

(Paar neu aufgestellt und gesehen, dass es zusammenführbar ist.)

Wie überprüfen wir denn nun, ob die Gleichung $m(x, x) \equiv 0$ gilt. Die ist ja sinnvoll.

Wir reduzieren beide Seiten der Gleichung mit dem TES zur Normalform, das ist hier schon gegeben, und stellen dann fest, dass die nicht gleich sind, also die Gleichung nicht gilt.

Wie jetzt? Aber das ist doch richtig!

Ja, für natürliche Zahlen. Die haben wir ja auch im Kopf gehabt, als wir das Gleichungssystem aufgestellt haben. Die Gleichungen behandeln ja nur Fälle, in denen sich Terme durch m, s und 0 darstellen lassen. Also genau die Grundterme. Wenn wir in die Trägermenge der Algebra noch zusätzliche Objekte einbauen, dann ist die Gleichung unter Umständen nicht mehr richtig.

Was wollen wir also?

Das die Gleichung für alle Grundinstanzen gilt.

Genau. Induktive Gültigkeit. Wie geht das denn?

Die Konsistenzbeweismethode sagt uns, dass eine Gleichung genau dann induktiv gültig ist, wenn die Hinzunahme der Gleichung keine neuen Grundgleichungen wahr macht. Wenn wir jetzt ein äquivalentes, konvergentes TES haben, dann ist die Aussage äquivalent zu: Die Hinzunahme der Gleichung macht keine zwei verschiedenen Grundnormalformen des TES gleich. Und wenn das TES zusätzlich noch das Definitionsprinzip erfüllt, dann ist das gleichbedeutend zu: Die Hinzunahme der Gleichung macht keine zwei Konstruktorgrundterme gleich. Das können wir dann automatisch überprüfen, indem wir eine zusätzliche Regel für Inkonsistenzen in den erweiterten Vervollständigungsalgorithmus einbauen.

Richtig. Machen sie dass doch auch noch eben.

Also erstmal die Gleichung orientieren zu $m(x, x) \to 0$.

Was muss man da beachten?

Es muss $m(x, x) \succ 0$ mit der gleichen Reduktionsrelation wie für die anderen Regeln gelten und die linke Seite muss außen ein definiertes Funktionssymbol haben.

Okay. Weiter?

Dann bilden wir wieder kritische Paare. Wenn wir die neue Regel mit den ersten beiden überlagern... Muss ich das machen? Die sind zusammenführbar.

Wenn sie das so sehen ist das okay.

Also nur das dritte kritische Paar. (Das habe ich wieder falsch aufgestellt. Sollte mir die Definition nochmal ansehen und dann nochmal aufstellen. Dieses mal richtig.) Da können wir jetzt Reduziere-Gleichung anwenden und dann die Gleichung löschen.

Genau. Jetzt sind wir mit dem Algorithmus durch. Was wissen wir denn nun?

Das die Gleichung $m(x, x) \equiv 0$ induktiv gültig ist.

Ja. Wenn sie jetzt mal an Induktion denken - mit Induktionsanfang und Induktionsschluss und so - wo haben wir das denn hier in der impliziten Induktion?

???

Okay, anders. Wie zeigt man die Aussage denn mit Induktion.

Induktionsanfang für $x = 0$ und Induktionsschluss mit $m(s(x), s(x)) \equiv 0$ wobei die Induktionshypothese uns sagt, dass $m(x, x) \equiv 0$ schon gilt.

Richtig. Haben sie das hier gemacht?

Äh... Nein?

Hmmm. Was war denn das erste kritische Paar, dass sie jetzt nicht gebildet hatten? Wie sah die Regel dazu aus.

Die instantiierte Regel war $m(0, 0) \to 0$. Das entspricht dann dem Induktionsanfang.

Und für das letzte kritische Paar?

$m(s(x), s(x)) \to 0$. Das ist der Induktionsschritt.

Und wie ließ sich das zusammenführen?

Mit $m(x, x) \to 0$. Also genau der Induktionshypothese.

Ja. Sie sehen: Das ist keine Zauberei. Das stimmt wirklich. Aber machen wir mal mit Funktionale weiter. Schreiben sie mal ein Programm auf, dass die unendliche Liste der natürlichen Zahlen generiert.

$inf\ = 0 : map\ (+1)\ inf$

Ja. Das haben Sie ja direkt mit ho-Funktion gemacht. Schön. Und jetzt eins, dass eine Liste von Listen konkateniert.

$concat = foldr\ append\ [\,]$

Okay. Wie ist denn jetzt die Semantik von inf? Das ist ja wenigsten noch rekursiv.

Das ist der kleinste Fixpunkt der ho-Funktion, die die rechte Seite der Gleichung in die linke Seite überführt.

Wie sieht die hier aus?

$f\ inf = 0 : map\ (+1)\ inf$

Okay. Was ist der Typ davon?

$[Int] \to [Int]$

Richtig. Ist das eine ho-Funktion?

Nein.

Wann wäre es denn eine?

Wenn inf noch ein Argument bekommen würde.

Genau. Gibt es immer genau einen Fixpunkt?

Nein, es kann mehrere geben. Zum Beispiel hat die ho-Funktion zu $n\ x = n\ (x + 1)$ alle konstanten Funktionen als Fixpunkt.

Okay. Welchen nehmen wir dann?

Den kleinsten.

Existiert der immer?

Bei stetigen Funktionen: Ja.

Wie zeigt man das denn?

Das sagt und der Fixpunktsatz. Also vorausgesetzt wir haben einen Domain mit cpo.

Schreiben sie die Aussage des Fixpunktsatzes mal auf.

Wenn $\sqsubseteq_D$ eine cpo ist und $f : D \to D$ stetig, dann existiert der $lfp\ f = \sqcup\{f^i(\perp)|i \in N\}$.

Zeigen sie mal, dass $\sqcup\{f^i(\perp)|i \in N\}$ existiert.

Wir müssen nur zeigen, dass $\{f^i(\perp)|i \in N\}$ eine Kette ist. Dann folgt das auf der cpo. Also zeigen wir $f^i(\perp) \sqsubseteq f^{i+1}(\perp)$. Durch Induktion über i:
$i = 0 \quad f^0(\perp) = \perp \sqsubseteq f^1(\perp)$
$i > 0 \quad f^{i-1}(\perp) \sqsubseteq f^i(\perp)$ nach Induktionshypothese. Da f stetig ist und alle stetigen Funktionen monoton sind, gilt auch $f(f^{i-1}(\perp)) \sqsubseteq f(f^i(\perp))$. Also ist das eine Kette.

Warum sind alle stetigen Funktionen denn monoton? Kann man das zeigen?

Also zu zeigen ist dann: Wenn $d \sqsubseteq e$, dann auch $f(d) \sqsubseteq f(e)$. Wenn aber $d \sqsubseteq e$ gilt, dann ist $\{d, e\}$ eine Kette und $e = \sqcup\{d, e\}$. Dann ist $f(e) = f(\sqcup\{d, e\})$ und da f stetig ist, auch $f(\sqcup\{d, e\}) = \sqcup\{f(d), f(e)\}$. Damit ist dann offensichtlich $f(d) \sqsubseteq f(e)$ und f monoton.

Okay. Warum sind unsere Funktionen denn stetig?

Naja. Einen formalen Beweis dazu hatten wir in der Vorlesung nicht, aber wir haben uns ja auf stetige Funktionen beschränkt, weil nur die berechenbar sind. Aber alles, was wir als HASKELL Programm schreiben können ist ja berechenbar und somit stetig.

Kommen wir mal zum Lambda-Kalkül. Wie geht das mit dem Fixpunkt denn da?

Mit einer Konstanten fix und der δ-Regel $fix = \lambda f.f\ (fixf)$.

Und warum nicht $fix\ f = f\ (fix\ f)$. Also ohne das Lambda. Das ist doch viel besser zu lesen!

Wir haben aber nur bestimmte δ-Regeln zugelassen, damit die Reduktionen konfluent bleiben. Eine Bedingung war, dass die Terme geschlossen sein müssen. Das ist hier nicht mehr so.

Bei TES haben wir ja einfach nur gezeigt, dass kritische Paare zusammenführbar sind. Wäre das denn hier keine gute Idee.

Nein. Nicht gut. Die Zusammenführbarkeit kritischer Paare bedeutet nur lokale Konfluenz. Die ist aber nur äquivalent zu Konfluenz, wenn unsere Reduktionsrelation fundiert ist. Das ist sie hier nicht.

Zum Beispiel?

bot $\to$ *bot*

Und fix?

Das terminiert auch nicht.

Wenn man jetzt aber keine Konstanten haben will, was kann man denn da machen?

Im pure-Lambda kann man den Fixpunktkombinator so darstellen: $\lambda xy.$ ...

Na lassen sie das mal. Es geht auf jeden Fall. Warum nehmen wir denn nicht den?

Weil der nicht mehr typkorrekt ist, da er eine Selbstapplikation enthält.

Also so etwas wie $x\ x$. Ist das nicht typkorrekt?

Nein. (Stimmt aber nicht - das ist typkorrekt)

Nie?

Also zum Beispiel $(\lambda x.3)(\lambda x.3)$ würde gehen.

(Ich sollte dann den Typinferenzalgorithmus auf $x\ x$ anwenden. Da ist mir dann auch aufgefallen, dass das korrekt ist.)

Was wäre denn nicht korrekt? Was müssten wir da ändern?

$\lambda x.x\ x$

Richtig. Dann machen wir jetzt Compilerbau. Welche Sprachklasse behandeln wir bei der lexikalischen Analyse.

Die regulären Sprachen.

Sind die unter Negation abgeschlossen?

Ja. Das ist aber aufwendig. Man muss erst eine $DTA(\alpha)$ konstruieren und dann die Endzustände invertieren.

Für NEAs geht das also nicht?

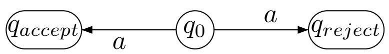
Nein.

Warum denn nicht? Haben sie da ein

Beispiel?

Nein, habe ich nicht.

(Dann hat mir Prof.Giesl ein Beispiel aufgemalt. Ich sollte dann die Endzustände umkehren.)



Was ist denn das erweiterte Matchingproblem?

Wir wollen für eine Menge an Regulären ausdrücken untersuchen ob es für ein Wort eine Zerlegung gibt, so dass die Teilwörter durch einen unserer regulären Ausdrücke beschrieben werden.

Ist die denn eindeutig?

Nein, im Allgemeinen nicht. Weder die Zerlegung, noch die Analyse, die wir daraus generieren wollen. Deshalb hat man bestimmte Konventionen. Für die Eindeutigkeit der Analyse wählt man den ersten Match und für die Zerlegung den längsten Match.

Wie bekommen wir den denn?

Wir konstruieren zu jedem regulären Ausdruck einen endlichen Automaten, bilden daraus den Produktautomaten und zerlegen die Endzustandsmenge wegen der *first match* Bedingung in disjunkte Teilmengen. Dann erweitern wir diesen Automaten zum Backtrackautomaten.

Und wie sehen die Transitionen von dem Ding aus?

Also wir haben die Zustandsmenge $(\Delta \uplus \{N\} \times \Sigma^*Q\Sigma^* \times \Delta \uplus \{error\})$.

Und die Transitionen? Wir kommen wir denn nun wenn wir vorne ein N haben in den Backtrack-Modus?

Also wenn $\delta(q, a) = q'$ ist und wir eine Konfiguration (N, qaw, W) haben, dann ist das Ergebnis $(T_i, q'w, W)$ wenn q' ein Endzustand ist, *error* wenn er unproduktiv ist und $(N, q'w, W)$ sonst. Im ersten Fall sind wir dann im Backtrack-Modus und suchen nach längeren Matches, wobei wir die gelesenen Zeichen, dann vor das q schreiben, weil wir ja nicht wissen, ob da noch ein längerer Match kommt.

Okay. Was untersuchen wir denn bei der Syntaxanalyse?

Die kontextfreien Sprachen.

Sind die denn unter Komplement abgeschlossen? Bei den regulären Sprachen wissen wir das ja jetzt.

Ich würde vermuten, dass Sie es nicht sind, aber warum...

(Habe ich nicht mehr alles im Kopf, aber es ging darum, dass die Menge der Sprachen, die von deterministischen Kellerautomaten erkannt wird eine echte Teilmenge von den nicht-deterministischen Kellerautomaten ist und dass letztere nicht unter Komplement abgeschlossen sind.
Die genauen Fragen die dann kamen weiß ich auch nicht mehr. Letztendlich sind wir dann auf den CYK-Algorithmus gekommen und dass der nicht so schön ist, weil der kubische Laufzeit hat.)

Statt dessen betrachten wir spezielle Grammatiken wie $LL(k)$ oder $LR(k)$.

Wie ist das denn hier mit der Zeit?

Die lassen sich in linearer Zeit erkennen.

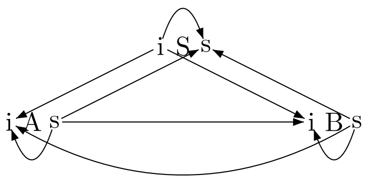
Was ist den $LL(k)$?

Eine Grammatik ist $LL(k)$ wenn für alle Linkssatzformen gilt, dass aus $S \Rightarrow_l^* wA\alpha$, $A\alpha \Rightarrow_l \beta\alpha \Rightarrow_l^* x$, $A\alpha \Rightarrow \gamma\alpha \Rightarrow_l^* y$ und $first_k(x) = first_k(y)$ folgt, dass $\beta = \gamma$ ist. Wir können also anhand der nächsten k Symbole eindeutig sagen, welche Regel wir anwenden müssen.

Und wir finden wir jetzt raus ob eine Grammatik z.B. $LL(1)$ ist?

Bei $LL(1)$ indem wir die look-ahead Mengen berechnen, also $la(A \to \alpha) = fi(\alpha\ fo(A))$ und für Regelalternativen $A \to \alpha|\beta$ gucken, ob die entsprechenden look-ahead Mengen disjunkt sind.

Genau. Machen wir mal Attributgrammatiken. Wenn wir eine Regel $S \to AB$ haben und jedes Nichtterminal ein synthetisches und ein inherites Attribut hat, was für Abhängigkeiten kann es dann geben?



Und wie entstehen da Zykel?

Hier gar nicht. Das wird auf lokaler Ebene durch die Einteilung in Innen- und Außenattributen verhindert. Da müsste dann noch ein Knoten drunter. (Knoten drunter gemalt und Zykel gezeigt.)

Okay. Wir verhindern wir so etwas?

Durch spezielle Grammatiken, etwa S- oder L-Grammatiken.

Was ist denn bei L nicht erlaubt?

Also wenn ein inherites Attribut eines Unterknoten von einem synthetischen eines Unterknoten abhängt, dann muss dass synthetische weiter links stehen.

Okay. Dann lassen sie uns mal kurz alleine.