Pruefungsprotokoll
Vertiefung: Implementierung von Programmiersprachen
Faecher: Compilerbau, Programmanalyse und Compileroptimierung, Logikprogrammierung
Pruefer: Indermark
22.04.2005
Note: 1.0

Compilerbau

Fangen wir mit Compilerbau an. Kann man denn den Automaten, der für die lexikalische Analyse zuständig ist, noch minimieren bezüglich der in der Vorlesung vorgestellten Konstruktion?

Endliche Automaten kann man minimieren, indem man die äquivalenten Zustände zusammenfasst (Prof. Indermark hat dabei geholfen und dann selbst gesagt, dass man da beim Produktautomaten aufpassen muss, weil es ja nicht nur interessant ist, dass der Automat in einen Endzustand gelangt, sondern welche Komponente des Endzustandes die entscheidende ist. Diese soll ja dann ausgegeben werden)

Kommen wir zur syntaktischen Analyse: TD-Analyse. Wie kann man da entscheiden, welche Regel angewendet wird?
la-Mengen bilden, Alternativen auf Disjunktheit prüfen (Habe da ein Beispiel aufgeschrieben)

Wann ist denn ϵ in la(Π_i)?
wenn es in fi($\beta fo(A)$) ist.

Wann kann das denn passieren?
Beschrieben, was fi und fo ist und was das in diesem Zusammenhang bedeutet.

Wenn man jetzt eine Regel hat: $A \rightarrow B$. Wie ist denn dann der Zusammenhang zwischen fo(A) und fo(B)?
Alles, was in fo(A) ist, ist auch in fo(B). Beschrieben warum.

Kommen wir zur BU-Analyse: schreiben Sie mal auf: [A $\rightarrow$ $\alpha\cdot$], [B $\rightarrow$ $\beta\cdot$]. Was kann man denn hier über α und β aussagen?
Sie müssen das gleiche Suffix haben.

Kommen wir zur semantischen Analyse. Codeerzeugung mit Attributen. Wie geht das? Zum Beispiel für die Produktion $E \rightarrow E + E$
c.0 = c.1 ; c.3 ; add. Code als synthetisches Attribut benutzen. Habe ich noch ge-

nauer erklärt.

PACO

Wir haben ja verschiedene Analysen kennengelernt. Wie ist denn das bei der Dead-Code-Elimination?
Es gibt LV- und NV-Analyse. Beide sind Rückwärtsanalysen. Die LV-Analyse funktioniert so:..., die NV-Analyse funktioniert so:... . Die NV-Analyse ist also „besser“.

Und wie ist das dann mit der Optimierung?
Die Optimierung,die auf den NV-Mengen basiert, ist „besser“, weil man möglicherweise mehr rausschmeißen kann.

Kann man denn nach einem Durchlauf der auf LV beruhenden Dead-Code-Elimination noch weitermachen?
Ja, man kann möglicherweise nochmal etwas eliminieren.

Wie ist das bei NV?
Da geht das nicht.

Gut. Wir hatten ja Flußdiagramme und -graphen. Was ist denn da der Unterscheid?
Die Diagramme benutzt man für die semantische Analyse. Da interessiert dann natürlich auch, unter welchen Umständen man welche Abzweigung nimmt. Beim Flußgraphen wird davon abstrahiert, denn man ist an einer Approximation der Analyseinformationen interessiert. Die Programmpfade sind regulär, die Berechnungspfade sind rekursiv aufzählbar, aber im Allgemeinen nicht entscheidbar.

Wie macht man das denn mit der semantischen Analyse?
Man bestimmt die Fortsetzungsfunktionen. (Beispiel angegeben und erläutert und Fixpunktiteration angesprochen.

Gut, dann kommen wir doch mal zu den Fixpunkten: Wieso interessieren wir uns dafür und wie berechnet man die?
Wir wollen wissen, was das Programm tut bzw. eine möglichst gute Abschätzung der Analyseinformationen für IC-Programme bekommen. Bei ACC fallen Monotonie und Stetigkeit zusammen, da der join dann einfach das größte Element ist. Und wegen Monotonie muss der join des größten Folgegliedes auch der größte Funktionswert sein. Das ist dann genau die Definition von Stetigkeit.

Und wie berechnet man den Fixpunkt dann?

$\perp \leq f(\perp) \leq f(f(\perp))\ldots$ = fix(f). Das ist dann gerade der join.

Warum gilt die zweite Ungleichheit?
Weil $\perp$ kleiner als $f(\perp)$ ist und die Funktion monoton ist.

Logikprogrammierung
Wie sind denn die SLD- und die lineare Resolution definiert?
Definition angegeben.

Wie groß ist denn dann so ein Resolvent?
Die Anzahl der Literale kann sich nur um 1 verringern (wegen binärer Resolution).

Beschreiben Sie doch mal so einen Ableitungsschritt

Da Sie ja jetzt schon den SLD-Baum angesprochen haben: Wie ist denn das bei Datalog?
Da kann es auch unendliche Pfade geben. Die kann man dann aber erkennen.

Wieso? Was ist denn überhaupt ein Pfad?
Konfigurationen erklärt, wie man von einer zur anderen kommt, usw.

Und warum kann man da jetzt die unendlichen Pfade erkennen?
Weil man keine Funktionen hat. (Da stand ich ein wenig auf dem Schlauch, er fragte dann danach, was ein Literal ist. Totaler Aussetzer. Aber ich bin dann irgendwann auch drauf gekommen und konnte mich dann erinnern an mein Skript: Wiederholung mit gleichem Gesamtzustand, Wiederholung mit wachsendem Gesamtzustand)

Das ist noch etwas komplizierter, da habe ich in der Voprlesung aber nichts mehr zu gesagt...

Natürlich ist das nicht der Original-Wortlaut. Aber so ungefähr ist es abgelaufen. Man sollte sicherlich nicht nur immer die Fragen beantworten, sondern es empfiehlt sich, auch mal selber ein bisschen weiterzudenken. Das zeigt ja auch Interesse an der Materie. Aber vorsicht: nicht über das Ziel hinausschießen (das kann ich aus eigener Erfahrung sagen). Insgesamt kann man sagen, dass es eine sehr entspannte Prüfungssituation war. Für mich war diese Prüfung eine Wiederholung und ich habe mir den Rat von Professor Indermark zu Herzen genommen, dass es keine Schwäche ist, ersteinmal über jede Frage kurz nachzudenken, auch wenn man denkt, dass man die Antwort direkt sagen könnte. Wenn eine Frage gestellt wird, die man so nicht direkt beantworten kann, ist das auch gar nicht schlimm. Es ist mir auch passiert. Wenn man dann ruhig bleibt und Professor Indermark auch seine Gedanken mitteilt, dann entsteht ein „nettes Gespräch“ und die Zeit geht wirklich super schnell vorbei. Ich habe mir im Vorfeld viel zu viele Gedanken darüber gemacht, dass Definitionen abgefragt werden, weil man das immer so mitbekommen hat. Klar, la-Mengen und so sollte man drauf haben. Viel Erfolg allen, die es noch vor sich haben! Professor Indermarrk ist ein super Prüfer und ich empfehle allen, die die Möglichkeit haben, sich noch bei ihm Prüfen zu lassen.