Prüfungsfragen Algorithmische Sprachen 1999

Syntaxanalyse
Parallele Algorithmen
Schneider, Beisitzer: Gabriella Kókai
April 1999

Bemerkungen zu Prüfung und Prüfer

- Ergebnis: 1,7
- Prüfer freundlich aber distanziert (kein Shakehands zu Beginn)
- Gibt Hilfestellung ohne gleich die Antwort zu liefern
- Fragen sind weitgehend konkret
- Zur Vorbereitung reichen Vorlesung und Übung

Fragen

Syntaxanalyse

- Tabellen-/Ableitungsorientierte Verfahren
- Earley-Verfahren
  - Predictor-, Scan-, Completer-Schritt
  - Definition der E_{ij}
  - Laufzeitverhalten *($O(n^3)$ mit Standardalgorithmus)*
- Top Down und Bottom Up Verfahren allgemein *(Skizze)*
- $LL(k)$-Analyse
  - Ableitungsklassen versus Oberklassen
  - Warum nicht LL(0)? *(endliche Sprache)*
  - Wie sieht die Tabelle aus?

Parallele Algorithmen

- List Ranking
  - Problemstellung *(allgemein, nicht nur für Suffix Sum)*
  - Wie funktioniert das Pointer jumping?
  - Warum ist der Algorithmus nicht optimal und was heißt das?
  - Wie kann er verbessert werden *(nicht Brent, sondern die Symmetry breaking Verfahren)*
  - Was sind r-Ruling-Sets?
  - Wie funktioniert das mit $logn$-Ruling-Set?
  - Was erhält man dabei? *(Sublisten der Länge $n/logn$)*

Syntaxanalyse, Compilerbau, Grundlagen Objekt-Orientierter Softwarenentwicklung (GOOSE)
PD. Dr. Wilke, Beisitzer: Dr. Minas
Juli 1999

Bemerkungen zu Prüfung und Prüfer

- Ergebnis: 1,3
- Zur Prüfung:
  Schneider war krank (Horror!) aber Wilke bietet sich als Alternative an. Wer ist das? Aber egal, es soll vorbei sein. :-) Beide (besonders Minas) geben sich viel Mühe, die Situation zu entspannen, machen Witze etc. - erfolgreich. Man kann als Prüfling die Richtung der Prüfung steuern; das war mir nicht in letzter Konsequenz klar, sonst hätte ich mich nicht leichtfertig auf Earley eingelassen. Bewertung war fair. Am Ende ausführliche Begründung der Note und Tips für erfolgreiche Prüfungen.
- Zur Vorbereitung:
  Es war wohl absolut nicht erwartet, daß man die ganzen Papers liest, die da immer ausgeteilt werden. Skript reichte völlig aus, auch wenn ich den Wilhelm/Maurer: Übersetzerbau und in Teilen den Tremblay/Sorenson: Compiler Writing wirklich empfehlen kann, weil Schneiders Folien da in Prosa drinstehen. Die Übungen in SA sind wohl nur insofern wichtig, als sie der Illustration der Vorlesung dienen. Was darüber hinaus geht, kann man getrost weglassen. Das gleiche gilt wohl für Beweise, obgleich die ein oder andere formale Darstellung/Definition sicher mal vorkommen kann.

Syntaxanalyse

- Syntaxanalyse allgemein:
  Was ist die Problemstellung, wie geht man das an (wirklich 5 Minuten so allgemeines Zeug, Wortproblem formal, Ableitbarkeit, direkte Ableitbarkeit, etc. – darauf war ich nicht gefaßt)
- ableitungs- vs. tabellenorientierte Verfahren
- Verfahren von Earley (Dreiecksgraphik, die Schritte, Verfahren skizzieren, wann geht man in welches Matrixelement etc.)
- Effizienzüberlegungen (Earley O(n^3), warum? etc.)
- Vorteil von LL(k) und LR(k) gegenüber Earley
- Warum ist Earley trotzdem noch wichtig? (Man hat nicht immer LR(k)- Eigenschaft – z.B. in Compilerbau-VL: Codegeneratorgeneratoren)
- Voraussetzungen an Grammatik bei den Verfahren
- Warum ist bei LR(k) k = 1 so interessant? Was sind die Probleme bei k > 1?

Compilerbau:

Da gab es direkt gar keine Fragen. Er hat das mit Syntaxanalyse und GOOSE kombiniert.

Grundlagen Objekt-Orientierter Software-Entwicklung (GOOSE):

(Minas hat vor der Prüfung noch schnell den Ordner durchgelesen, damit er weiß, was er fragen soll – ja, Minas hat diesen Teil gefragt)

- Was wurden für OO-Sprachen besprochen? (C++, Java, Eiffel, Smalltalk)
- Was sind die Unterschiede zwischen C++ und Java?
- Warum ist Mehrfachvererbung in Java verboten (offizieller Grund: Effizienz, inoffizieller: es ist schwierig im Compiler zu machen)
- Welche Arten von Vererbung gibt es (technisch gesehen): einfache, mehrfache abhängige, unabhängige
- Warum ist die Effizienz bei mehrfacher V. schlechter? (Da mußte ich des ganze Graffel über die Übersetzung von OO-Sprachen erzählen: Methodentabellen, Views, Substituierbarkeit, wie die Verzeigerung ist, was steht in der Methodentabelle, etc.)
- Auflösung von Namenskonflikten bei Mehrfachvererbung in den verschiedenen o.g. Sprachen: der erste in der Liste, explizit, verbieten, ...
- Warum muß der Zeiger auf die MT immer am Anfang des Objekt-Speicherbereiches stehen?

Graphtransformationssysteme, Programmierung der Graphtransformationssysteme, Ergänzungen zur Syntaxanalyse (Compilerbau)
Professor Dr. Schneider, Beisitzer: Harald Schmidt
Oktober 1999

Bemerkungen zu Prüfung und Prüfer

- Ergebnis: 1,7
- Prüfer fragt zunächst, mit welcher Vorlesung begonnen werden soll
- gibt Hilfestellungen
- fragt nach Beweisen oder Definitonen der wichtigsten Konzepte
- verlangt eher die groben Züge und das wesentliche Kernstück eines Beweises
- erwartet eher eine anschauliche Erklärung eines Beweises
- eher Verständnis als Detail

Fragen

Graphtransformationssysteme

- Definition des Pushouts

- Eindeutigkeit des Pushouts (analog zu Eindeutigkeit des Coprodukts, Vorl. 3.3)
- Doppelpushoutkonstruktion und Ableitbarkeit
- Wie bekommt man den Kontext C? (durfte selber entscheiden, für welchen Fall → Coprodukt-Komplement-Konstruktion

Programmierung der Graphtransformationsysteme (Miranda und Java)

- fragte zunächst, ob Miranda oder Java lieber als Thema ist (entschied mich für Miranda)
- Wie haben wir Kategorien definiert? (algebraischer Datentyp)
- Wie sieht der Typ für Pushout aus?
  meine Antwort: (**,**) → (**,**,(**,**)→**)
- Wie sieht das in Schreibweise für Curryfunktionen aus? (unleserlich, **→**→...

Ergänzung zur Syntaxanalyse (Compilerbau)

- Stackframe bei zuweisungsorientierten Sprachen
- Was sind static link bzw. dynamic link (verweisen auf Datenbereich der im statischen Sinne umgebenden Prozedur bzw. auf Aufrufer)
- Wozu haben wir static link bzw. dynamic link (Zugriff auf globale Größen bzw. Rücksprung)
- Wofür ist der extreme pointer?
- Stackorganisation bei funktionalen Sprachen (Besonderheit: Curryfunktionen und Adressierung relativ zu SP mittels sl)

Syntax, GOOSE, Compilerbau, Prof. Schneider, Beisitzerin: G. Kokai Oktober 1999

Bemerkungen zu Prüfung und Prüfer

- Ergebnis: 1,7
- Die Fragen kamen immer recht „kontexfrei“, ohne große Einleitung (was aber auch Tagesform des Prüfers sein kann).
- Zum Teil ziemlich schnell ziemlich tief ins Detail, manchmal „retten“ einen aber auch schon Stichworte.
- Sehr faire Benotung mit Begründungservice.

Fragen

Syntax

- (-kein allgemeines Überblickswissen, schade :-)
- Was ist der Unterschied zwischen tab.orient und abl.orient Verfahren?
- Effizienz?
- Woher die Namen, tab.orient und abl.orient, wenn doch beide Tabellen brauchen?
- Earley, ziemlich detailliert
- LL(k) mit genauer Erklärung, was man da hinschreibt
- Ableitungsklassen, Oberklassen mit Definition

Compiler

- Was kommt bei Funktionalen Sprachen auf den Stack?
- Was bei Prozeduralen?
- Wozu Dynamischer/Statischer Link?
- EP-Register

Goose(Grundlagen OO-SW-Entwicklung)

- Überblick (endlich!)
- Problem der Mehrfachvererbung
- Wie in Java?

Graphentransformationsysteme, Syntaxanalyse, Prof. Schneider, Beisitzerin: G. Kokai Oktober 1999

Bemerkungen zu Prüfung und Prüfer

- Ergebnis: 1,0

Fragen

GTS

- Was ist ein Pushout?
- Wie zeigt man die Eindeutigkeit des Pushouts? (Nichtwissen hat nicht gestört)
- Was versteht man unter dem Doppelpushout-Ansatz?
- Wann existsiert ein PO-Komplement? Wie konstruiert man es (bei injektiver Einbettung, Klebebedingung!)?
- Wie ist die parallele Unabhängigkeit definiert?
- Was versteht man unter kontextfreier Hyperkantenersetzung (Def. der kf. Hyperkantenproduktion)?
- Sind kf. Hyp.kantengrammatiken mächtiger als kf. Chomsky-Gr.? (ja: $a^n b^n c^n$)

Syntax

- Unterschied Top-Down- und Bottom-Up-Verfahren
- Def. Ableitungsklasse
- Warum Oberklassen? (Entscheidbarkeit der Disjunktheit)
- Def. LL(k) Oberklassen
- Was bekommt man, wenn man auch den Stackinhalt begrenzt? (Starke LL(k) Oberklassen)
- Def. Reduktionsklasse
- Wozu braucht man die Shift-Klasse, wie ist sie definiert?
- Wie erhält man die LR(0)-Zustände (sehr knappe Erläuterung reichte)?
- Wie groß ist der Aufwand bei Erstellen der LR(1)-Tabellen? (exp.)
- Warum ist das nicht schlimm? (Tabelle nur einmal pro Grammatik, Parsing dann linear)