Prüfungsfragen Algorithmische Sprachen 2004

Compilerbau, Maschinelles Lernen
Prof. Dr. Michael Philippsen, Gabriella Kokai
April 2004

Bemerkungen zu Prüfung und Prüfer

- Ergebnis: 1.0
- Der Compilerbau-Teil wurde von Prof. Phillipsen und der Machine Learning Teil wurde von Gabriella Kokai geprüft.
- Prof. Phillipsen fragt ins Detail und Verständnis gleichzeitig. Ich hab eigenlich kaum in Büchern nachgelesen, stattdessen genug schlaue Leute gefragt :-)
- Die Gabi fragt vor allem Verständis und lässt einen selbst bestimmen wie weit man ins Detail gehen will. Fragt nach, wenn sie es genauer wissen will. Laesst einem die Möglichkeit die Prüfung in die Richtung zu lenken die man will. Es ist sinnvoll den Mitchell durchzuarbeiten, weil das Skript an manchen Stellen nicht so leicht verständlich ist.
- Ich durfte mir aussuchen mit welcher Vorlesung ich anfangen will.

Fragen

Compilerbau

- Was liegt in einem Stackframe?
  - Parameterübergabe, entweder auf Stack oder in Registern (flüchtende Register)
  - Rückgabewerte entweder auf Stack oder in Registern.
  - Callee und Caller save Register
  - Framepointer. warum nötig?
    → frühe Addressierung der Variablen mgl.,
    → Stack von beiden Seiten zugreifbar,
    → ein dynamisches Array auf Stack.
    Auf Nachfrage hab ich dann erklärt wie dynamische Arrays addressiert werden.
  - Statischer Vorgängerverweis. Verzeigern und display erkläert.
    Warum ist der noetig?
    Unterschied zu statischem Vorgängerverweis?
- Warum verwende ich Register, wenn man doch alles auf dem Stack sichern muss?
  → Blattprozeduren
- An welchen Stellen macht ein Compiler Fehlermeldungen?
  → Analysephase
  - lexikalische Analyse: verwendet Reguläre Ausdrücke/ Automaten, wenn ein Ausdruck passt wird ein Token erzeugt, wenn nicht Fehler.
  - syntaktisce Analyse: mit Grammatik wird Strukturbaum erzeugt, wenn keine Produktion passt → Fehler.
  - Woher weiss man wo der Fehler passiert ist? → an Tokens wird die Zeilennummer gespeichert.
  - Wie kann ich das realisieren wenn ich nicht nach jedem Fehler abbrechen will, sondern mehrere Fehler gleichzeitig ausgeben will?
    → Sich eine Stelle suchen an der man wieder einsteigen kann, zb. nach Strichpunkt, oder bei neuer Funktion, ...
  - sematische Analyse: Prüfung der Deklariertheitseigenschaft (Definitionstabelle), Typprüfung (Prototypen), allgemeinsprachliche Regeln/ prüfungen, die bei der syntaktischen Analyse nicht durchgeführt werden konnten (zb Arrayzugriff/Arraydecl).
- Wie Compiliere ich Goto? → Label und Sprung. Problem beim herausspringen aus Blöcken, da nicht bekannt ist vieviel Platz für Variablen freigegeben werden muss. Deshalb ist Goto nur innerhalb von Bereichen sinnvoll, für die nicht explizit Platz mehr für Variablen reserviert wird. (zB. bei Java innerhalb einer Funktion)

Maschinelles Lernen

- Beschreibt ein Problem mit bewerteten Beispielen aus Attributen. Wie kann ich aus den Daten lernen? → Induktives Lernen beschreiben
- Beschreibe einen Algorithmus mit dem das möglich ist. → ID3
  - Repräsentation der Daten und der Hypothese (Baum)
  - Grundlegenden Algorithmus
  - Reelle Attribute
  - Was tum bei fehlenden Attributen?
- Wie koennen Daten fehlerhaft sein? → falsch klassifiziert oder falsche Attributwerte.
- Was tun bei fehlerhaften Daten?
  → entweder Baumaufbau abbrechen, oder prunen (reduced error pruning, rule post pruning)
- Vorteile von reduced error pruning?
  → Regeln besser lesbar,
  → Attribute in Wurzelnähe können auch gepruned werden,
  → Zweige können unabhängig voneinander gepruned werden.
- Anderer Algorithmus um Regeln zu lernen?
  → CN2 (sequential covering, algorithmus ist specific to generell, neue Regel erzeugen ist generell to spezific)
- Unterschied zu FOIL?
  → CN2 ist beam search (mehrere Candidate hypotheses), FOIL ist greedy
  → CN2 wählt Regeln mit Entropie, FOIL mit Foil-Gain (daraus ergibt sich, dass der CN2 auch negative Regeln lernen kann, aber das war ihr glaub ich selbst nicht so ganz klar)
  → CN2 kann nur Aussagenlogische Regeln lernen, FOIL Regeln von Logik erster Ordnung.
- Unterschied zwischen Aussagenlogik und Logik erster Ordnung? → Logik erster Ordnung verwendet Variablen und ist deshalb Ausdruckstärker. (am Beispiel erklärt)

Compilerbau und Ausgewählte Kapitel aus dem Compilerbau
Prof. Philippsen, Beisitzer: Dominik Schell
September 2004

Bemerkungen zu Prüfung und Prüfer

- Schwerpunktfach
- Ergebnis: 1,0
- In meiner Prüfung wurde nur Verständnis gefragt - das Auswendiglernen aus dem Skript hätte ich mir, soweit ich es dann doch getan habe, sparen können. Die Fragen habe ich als schwer und fast etwas unfair empfunden, aber sie waren absichtlich so gestellt, um abzutesten, ob der Stoff wirklich verstanden wurde. Insgesamt hatte ich in der Prüfung selbst kein gutes Gefühl, aber im Nachhinein fand ich sie sehr fair. Compilerbau 1 und 2 wurden gemischt abgefragt, aber in anderen Prüfungen war dies anders.
- Vielleicht fällt es auch mit ins Gewicht, daß ich der letzte Prüfling an diesem Tag war.

Fragen

- Die erste Frage war, ob es Fragen geben würde, die auf keinen Fall gestellt werden sollen. Im Nachhinein war die Frage anscheinend tatsächlich ernst gemeint.
- Die erste richtige Frage war, wie der Stackframe aufgebaut ist, wobei dann etwas auf den dynamischen und den statischen Vorgängerverweis eingegangen wurde. (Z.B.: Woher kann man einen SVV bekommen, wenn man eine innere Methode direkt anspringt? Wozu ist der DVV gut?). Als nächstes kam die Frage, ob es in Java eine Entsprechung zu dieser Problematik gibt (innere Klassen) und wie das Problem dort gelöst werden kann.
- An das nächste Themengebiet wurde über ein paar Fragen zur Objektorientierung hingeführt, wobei die Fragestellung am Ende Richtung interprozeduraler Alias-Analyse ging. Die kr"öndende Abschlußfrage war, wie

man bei OO-Sprachen einen Prozeduraufrufgraphen aufbauen kann.

- Ein paar Minuten vor Schluß gab der Prüfer dann an den Beisitzer ab, der mir ein Blatt mit einem Datenflußbeispiel aus der Vorlesung vorlegte mit der freundlichen Bitte, dies auf Parallelisierbarkeit zu überprüfen.

Compilerbau 1 und Clustercomputing
Prof. Philippsen, Beisitzer Dr. Veldema
September 2004

Bemerkungen zu Prüfung und Prüfer

- Note: 1,3
- Philippsen prüfte hauptsächlich, Ronald Veldema stellte noch ein paar zusätzliche Fragen zu Clustercomputing.
- Beide Prüfer fragten zuerst allgemein, und stiegen dann auf Details ein.
- Nette Atmosphäre
- Alle Details, die gefragt wurden, basierten auf Verständnis (also zum Beispiel keine Frage nach Latenzzeiten oder so)
- In Clustercomputing genügte das Skript nicht zum Lernen, in Compilerbau schon (plus 1-2 Fragen an die Checker)

Fragen

Compilerbau

- Wo entstehen Fehler in einem Compiler?
  ⇒ Zum Einstieg erst einmal viel Reden, vor allem über die Analysephase, ich erwähnte aber auch die Methodenauswahl bei Polymorphie
- Semantische Analyse: Was geschieht da eigentlich?
  ⇒ Deklariertheitseigenschaft und Typkonsistenz prüfen
- Wie genau wird die Deklariertheitseigenschaft geprüft?
  ⇒ Top-Down, Stack, Hash, durchgefädelt...
- Wie genau wird die Typkonsistenz geprüft?
  ⇒ Bottom-Up, Tiefensuche, mehrere Durchläufe, evtl. Prototypen, das ganze Konzert...
- Wie ist die Auswertungsreihenfolge, muß man das nacheinander machen?
  ⇒ Hier mußte ich eklig genau sein. Zum Glück hatte ich das drauf.
- Wie geschieht die Methodenauswahl ohne Vererbung / Polymorphie etc.?
  ⇒ "Hartcodiert" reichte Philippsen nicht, er wollte auf den Strukturbaum hinaus und daß auch Methoden in der Namenstabelle stehen.
- Wie geschieht die Methodenauswahl mit Polymorphie?
  ⇒ Wähle Methode mit Kleinsten passender Typ, Eindeutigkeit sonst Fehler.

Clustercomputing

- Omega-Netz zeichnen
- `MPI_Reduce`-Operation implementieren - Allerdings nur den Algorithmus, nicht die exakten Methodenaufrufe mit Parametern etc.
- Was ist False Sharing?
  ⇒ Zwei Prozesse wollen auf die selbe Speicherseite schreiben.
- Was ist PBS?
  ⇒ Die Antwort habe ich bißchen mit SSI-Informationen im allgemeinen garniert.
- Was ist Active Messages?
  ⇒ Im Prinzip One-Way-RPC
- Wie funktioniert Schreibzugriff bei DSM?
  ⇒ Entweder beschafft sich der Schreiber exklusiv die entsprechende Seite, oder man verschmilzt die Änderungen mehrerer Schreiber. Das musste ich noch genauer fassen.