Universität Karlsruhe (TH)

Lehrstuhl für Programmierparadigmen

Sprachtechnologie und Compiler WS 2009/2010 http://pp.info.uni-karlsruhe.de/
Dozent: Prof. Dr.-Ing. G. Snelting snelting@ipd.info.uni-karlsruhe.de
Übungsleiter: Sebastian Buchwald sebastian.buchwald@kit.edu

Übungsblatt z | Ausgabe: 10.02.2010 | Besprechung: 10.02.2010

Auf diesem Blatt stellen wir die Themen der Vorlesung zusammen und geben einige exemplarische Fragen an. Die Fragen sollen weder besondere Relevanz für anstehende Prüfungen signalisieren noch das Fachgebiet vollständig überdecken.

1. **Architektur von Übersetzern**: Definition von Übersetzung, Korrektheit, Struktur, Module, Datenstrukturen

Analyse

2. **Lexer**: Datenstrukturen (Stringtabelle), Reguläre Ausdrücke, Endliche Automaten, Generatoren für Scanner
3. **Parser**:
   - Kontextfreie Grammatik, Konkrete / Abstrakte Syntax
   - LL: Rekursiver Abstieg, Vorschauproblem, Deterministisch-Machen, SLL
   - LR: Kellerklasse, Charakteristischer Automat, Konstruktion, SLR, LALR
   - Fehlerbehandlung: Syntax / Semantik, Systematische Behandlung
4. **Attributierte Grammatiken** (AG): Synthetisierte, Ererbte Attribute, Abhängigkeiten, Implementierungsansätze, Regeln für Typprüfung, Namensanalyse
5. **Semantische Analyse**: Namensanalyse, Typanalyse, Operatoridentifikation, Konsistenzprüfung, Sprachabhängige Sonderaufgaben, Strukturäquivalenz, Symboltabelle, Namenstabelle / Umgebung (Datenstruktur), Überladung vs. Polymorphie

Abbildung

6. **Transformation**: Zwischendarstellung (Tripel / Quadrupel / SSA), Grundblock, Typabbildung, Operatorabbildung, Ablaufsteuerung, Konturmodell, Speicherzuteilung, Parameterübergabeformen, SSA Definition, SSA Aufbau

Codierung

7. **Codeerzeugung**: Maschinensimulation, Zielattributierung, (Lokale) Registerzuteilung, Normalformtheorem, Codeauswahl(selektion), Makrosubstitution, Termersetzung (TES, GTES) Codeselektionsgeneratoren (BEG)

Aufgaben

1. **Architektur von Übersetzern**
   (a) Ihr Compiler läuft zu langsam, woran denken Sie zuerst?
   (b) Der Compiler darf jedes Programm in gewissen Grenzen umgestalten (optimieren), welche sind das?
   (c) Was braucht man für einen Ein-Durchlauf-Übersetzer?
   (d) Wie und wo werden doppelte Vereinbarungen erkannt?
   (e) Was ist eine Hashfunktion? Was bedeutet Kollisionsauflösung?
   (f) Auf welche Module des Compilers greift die Fehlerbehandlung zu?
2. **Lexikalische Analyse**
   (a) Wie wird ein NEA zu einem äquivalenten DEA? Was heißt hier äquivalent?
   (b) Was liefert der Lexer an den Parser?
   (c) Wozu braucht man eine Stringtabelle? Wie ist Sie aufgebaut?
3. **Syntaktische Analyse**
   (a) Was sind Situationen?
   (b) Kann man eine LL(1)-Akzeptor von Hand schreiben? Wenn ja, wie sieht das aus? Geht das vielleicht bei einem LR(1)-Akzeptor?
   (c) Wie lautet die SLL(1)-Definition?
   (d) Erklären Sie die Konstruktion eines LR(k)-Parsers.
   (e) Erklären Sie die Funktionsweise eines LR(k)-Parsers.
   (f) Warum ist LL(2) $\neq$ SLL(2)?
   (g) Was macht man mit einer Grammatik, die nicht LL(1) ist, man aber nur ein Werkzeug hat, das LL(1) behandeln kann?
4. **Attributierte Grammatiken**
   (a) Wozu dienen Attributierte Grammatiken?
   (b) Wie ist eine attributierte Grammatik definiert?
   (c) Wie sehen Attributierungsregeln aus?
   (d) Was sind ererbte und synthetisierte Attribute?
   (e) Was bedeutet LAG(1)?
5. **Semantische Analyse**
   (a) Was liefert der Parser an die semantische Analyse?
   (b) Wie wird der Gültigkeitsbereich von Variablen modelliert?
   (c) Wozu braucht man eine Symboltabelle?
   (d) Wie wird das Attribut "Umgebung" implementiert?
   (e) Wie führen Sie die Attributierung für die Bezeichnerzuordnung aus?
6. **Transformation**
   (a) Was macht die "Transformation"?
   (b) Was bedeutet Grundblock? Wofür ist er gut?
   (c) Welche Formen der Parameterübergabe kennen Sie?
   (d) Was passiert bei einem Funktionsaufruf?
   (e) Was ist ein Activation Record?
   (f) Wie kann man verschachtelte Funktionen behandeln?
   (g) Warum braucht man ϕ-Funktionen? Wo muss man Sie platzieren?

7. **Codeerzeugung und weitere Backend-Aufgaben**

   (a) Was ist die Grundidee / Technik der Codeerzeugung?

   (b) Wie bestimmt man die Anzahl der benötigten Register bei der Ausdrucksberechnung?

   (c) Was ist GTES (GTES=Grundtermersetzungssystem)?

   (d) Wie kann man aus einem AST Zielcode machen? Mehrere Möglichkeiten!

   (e) Was sind Register / Wertdeskriptoren?

   (f) Kann man auf die Gucklockoptimierung/Nachoptimierung verzichten?