Vorlesung Übersetzerbau (SS 2018)

Michael Hanus

2. Juli 2018

Die Übersetzung von Programmiersprachen ist eine wohldefinierte aber dennoch komplexe Aufgabe. Zur Beherrschung dieser Komplexität wurde eine Zerlegung in einzelne Übersetzungsaufgaben entwickelt, die heute in dieser oder ähnlicher Form in den meisten Übersetzern verwendet wird. In dieser Vorlesung werden die einzelnen Übersetzungsaufgaben und die bekannten Lösungsansätze dazu vorgestellt. Im Einzelnen werden behandelt:

- Programmiersprachen, Interpreter, Übersetzer
- Lexikalische Analyse
- Syntaktische Analyse
- Semantische Analyse
- Codeerzeugung

Detaillierter Vorlesungsverlauf

9.4.: **Einführung:** Übersetzer, Quellprogramm, Zielprogramm, Übersetzungsaufgabe, typische Übersetzerstruktur (lexikalische Analyse, syntaktische Analyse, Parsing-Aktionen, semantische Analyse, Zwischencodeerzeugung, Codeauswahl, Assembler), Frontend, Backend, Simple als einfache Beispielsprache, Darstellung von Ableitungsbäumen in Haskell

12.4.: **Programmiersprachen, Interpreter, Übersetzer:** Programmiersprache als partielle Funktion, Interpreter, Übersetzer, partieller Auswerter, Erzeugung von Übersetzern aus Interpretern mittels partieller Auswertung

16.4.: Kombination von Interpretern und Übersetzern, abstrakte Maschinen (JVM), Bootstrapping; Einführung in die **lexikalische Analyse**, Aufgabe eines Scanners, reguläre Ausdrücke

19.4.: Übersetzung in NFA, principle of longest match, ϵ-Abschluss, Konvertierung NFA in DFA, Hinweise zur Zustandsreduktion; Einführung in die **syntaktische Analyse**: kontextfreie Grammatik, Ableitungen, erzeugte Sprache, Ableitungsbaum, eindeutige und mehrdeutige Grammatiken

23.4.: Recursive-descent Parser, Implementierung in Haskell, (starke) LL(k)-Grammatik, Definition FIRST_k und FOLLOW_k, Steuermenge D_k, Spezialisierung fuer $k = 1$, Algorithmus zur FIRST/FOLLOW-Berechnung

26.4.: Konstruktion einer Parsing-Tabelle, Elimination von Linksrekursion, Linksfaktorisierung, Fehlerbehandlung in RD-Parsern; Einführung in die Bottom-Up/LR(k)-Analyse, shift-reduce-Parser, prinzipielle Arbeitsweise

30.4.: LR-Parsing-Tabelle, LR(0)-Element, LR(0)-Zustand goto-Funktion, Berechnung des LR(0)-Automaten, Berechnung der LR(0)-Parsingtabelle, LR(0)-Grammatik, SLR-Parser und SLR(1)-Grammatik

3.5.: LR(1)-Element, LR(1)-Abschluss und goto-Funktion, LR(1)-Parsingtabelle, LR(1)-Grammatik, Beispiel für LR(1)-Parsing-Tabelle (Dereferenzierung à la C), LALR(1)-Idee, LALR(1)-Parsingtabelle, LALR(1)-Grammatik, Zusammenhang (Inklusion) der Grammatik- und Sprachklassen; Parser-Generatoren am Beispiel von Happy

14.5.: Konflikte und deren Lösung, if-then-else-Konflikte, Parserdeklarationen für Assoziativitäten, Parserdeklarationen für Prioritäten; Semantische Aktionen, Implementierung in RD-Parsern (digits-Beispiel), Realisierung in Bottom-Up-Parsern (digits-Beispiel, Desk Calculator), Komplexität semantischer Aktionen in RD-Parsern (Ausdruckswerte berechnen), Interpreter für Simple-Programme als semantische Aktion, semantische Werte als Umgebungstransformation

17.5.: Simple-Interpreter mit Ausgabe; attributierte Grammatiken, Attributierung, Beispiel: Berechnung der maximalen Blockschachtelung, S-attributiert, L-attributiert, Implementierung L-attributierter Grammatiken in RD-Parsern; Abstrakte Syntax, Beispiel für Simple, Implementierung abstrakter Syntax in Haskell, Happy-Implementierung zur Generierung abstrakter Syntaxbäume für Simple

22.5.: Erweiterung um Positionsinformationen; Semantische Analyse, Scope-Verwaltung, Implementierung von Symboltabellen; Codeerzeugung (Überblick), Laufzeitspeicherorganisation: kellerartige Verwaltung von Prozeduren, Verwaltung von Blöcken, Speicherbereiche für dynamische Arrays und Records, Gesamtaufteilung des Laufzeitspeichers

28.5.: Aufbau von Prozedurrahmen, dynamischer und statischer Vorgänger, Übergabe der Rücksprungadresse in Register, Übergabe von Parametern in Registern, Verwaltung statischer Vorgänger, Display-Technik

31.5.: Probleme von Prozeduren als Parameter, Lambda-Lifting, Speicheraufbau für lokale Variablen, Records, statische und dynamische Felder, Vereinigungstypen, Funktionen als Variablen; Motivation für Zwischencode

4.6.: Stackmaschinencode, 3-Adresscode, abstrakte Ausdrucksbäume (Definition der Datentypen, Generierung temporärer Marken), Übersetzung in Zwischencode: l/r-Werte, Übersetzung von Konstanten, Variablen, Operatoren, Funktionen

7.6.: Übersetzung von Zeigervariablen, Zuweisungen, Konditionalen und while-Schleifen, Übersetzung von booleschen Ausdrücken (mit Sprüngen), Transformation abstrakter Ausdrucksbäume in Basisblöcke: Linearisierung, Basisblockgruppierung

11.6.: Umordnen von Basisblöcken zu Traces, Sprungcodeanpassung, Aufgabe der Zielcodauswahl Baummuster, Beispiele für unterschiedliche Zielcodauswahlen

14.6.: lokal optimale und optimale Zielcodes, Berechnung lokal optimaler Zielcodes (Maximal Munch), Berechnung optimaler Zielcodes mittels dynamischer Programmierung, Beispiel zur Berechnung optimaler Zielcodes; Lebendigkeitsanalyse: Motivation

18.6.: Lebendigkeitsanalyse: Kontrollflussgraph, Lebendigkeit von Variablen, Gleichungssystem, Fixpunktiteration, Berechnungsstrategien; Registerallokation für Basisblöcke: Informationsverwaltung (Registerinhalte, Variablenpositionen, Lebendigkeit)

21.6.: Auswahl eines freien Registers, sequentielle Codeinspektion, Beispiel für Registerallokation; Überblick über Techniken zur Code-Optimierung

25.6.: Algebraische Optimierung, Optimierung von Zwischendatenstrukturen partielle Auswertung, Konstantenpropagation, Konstantenfaltung, Kopierpropagation, Reduktion der Stärke von Operatoren, In-line expansion, Elimination redundanter Berechnungen

28.6.: Schleifenoptimierung, Verschiebung von Schleifeninvianten, Schleifenentfalten, Elimination toten Codes, Codeverschiebung über Basisblöcke, Peephole-Optimierung, Datenflussanalyse (use-def chaining, liveness), abstrakte Interpretation, Vorzeichenberechnung

2.7.: Implementierung logischer Programmiersprachen: Grundideen der WAM, prozedurale Implementierung, Heap-Speicher für Terme, Backtrack-Stack, Speicheraufbau