Universität Stuttgart

INSTITUT FÜR
KOMMUNIKATIONSNETZE
UND RECHNERSYSTEME
Prof. Dr.-Ing. Andreas Kirstädter

Bachelor-Arbeit / Forschungsarbeit Nr. 1085

Erweiterung eines Parsergenerators um Syntax-Error-Recovery-Mechanismen

Methoden
Programmierung
Softwareentwurf

Themengebiete
Compilerbau

Hintergrund

Am Institut entsteht aktuell eine experimentelle Variante der Programmiersprache Go mit dem Namen GoSUB. Für diese Sprache existiert ein Compiler, der selbst in GoSUB implementiert ist.

Im Frontend des Compilers liest zunächst ein Scanner den Quellcode ein und zerlegt ihn in Token. Daran schließt sich ein Parser an, der einen Syntaxbaum aufbaut. Abschließend folgen semantische Analyse und Zwischencode-Erzeugung.

Der Parser des Frontends ist von einem Parsergenerator erstellt, der von JavaCC inspiriert und in GoSUB implementiert ist. Der Generator liest eine Grammtik-Beschreibungsdatei ein und erzeugt daraus GoSUB-Code für einen Top-Down-Parser. Bislang unterstützt der Parsergenerator keine Syntax-Error-Recovery-Mechanismen. Ein generierter Parser bricht die syntaktische Analyse deshalb stets beim ersten Syntaxfehler ab.

Aufgabenstellung

Im Rahmen dieser Arbeit erweitern Sie den Parsergenerator, damit generierte Parser möglichst viele syntaktische Fehler in einem Durchlauf erfassen können. Erzeugte Parser sollen im Fehlerfall zunächst die sogenannte Phrase-Level-Recovery durchführen: Hier versucht der Parser, den Tokenstrom so zu modifizieren, dass er der Grammatik genügt. Dazu muss er Token finden, die er vor dem oder anstelle des Error-Tokens einsetzen oder entfernen kann. Diese findet er durch Betrachtung der Follow-Sets der Token und wählt den bestmöglichen Kandidaten aus. Schlägt dieser Fehlerkorrekturmechanismus fehl, soll der Parser die Analyse an einem Synchronisationspunkt fortsetzen.

Zuerst arbeiten Sie sich in die Sprache GoSUB und die bestehende Implementierung des Parsergenerators ein. Im nächsten Schritt ergänzen Sie die Parserbeschreibungssprache, um sprachspezifische Synchronisationspunkte anzugeben und Sie implementieren das Wiederaufsetzen des Parsers an den Synchronisationspunkten. Danach ergänzen Sie Ihre Implementierung um den Phrase-Level-Recovery-Mechanismus.

Erworbene Kenntnisse und Fähigkeiten

Sie erarbeiten sich die Grundlagen des Compiler-Designs und verstehen, wie zentrale Komponenten eines Compilers funktionieren. Darüber hinaus arbeiten Sie an einem großen Software-Projekt und erlernen eine neue, moderne Programmiersprache.

Voraussetzungen

Programmierkenntnisse

Kontakt

M.Sc. Tobias Schwientek
Raum 1.365 (ETI II), Telefon 685-69008, E-Mail tobias.schwientek@ikr.uni-stuttgart.de

Dipl.-Ing. Matthias Meyer
Raum 1.334 (ETI II), Telefon 685-67975, E-Mail matthias.meyer@ikr.uni-stuttgart.de