

Inhaltsverzeichnis

0. Compiler-Generierende Systeme	1
1. Definitionen und Grundlagen	8
1.1. Kontextfreie Grammatik	8
1.2. LR-Parser	10
1.3. Attributierte Grammatiken	12
2. Das Compiler-Generierende System POCO: Die Konzeption	21
2.1. Das Vorbild: MUGl	21
2.2. Zieldefinition für POCO	25
2.2.1. Überlegungen bzgl. der Generator-Struktur	27
2.2.2. Ein Typenkonzept für die semantischen Attribute ..	28
2.2.3. Das Modularitätskonzept auf der Ebene der generierten Compiler	29
2.2.4. Die Wahl einer geeigneten Implementierungssprache	31
2.2.5. Portabilität	33
3. Software-Portabilität	34
3.1. Das Übertragungsproblem	35
3.2. Portabilität unter Verwendung höherer Programmiersprachen	36
3.3. Compiler-Portabilität	43
3.3.1. Bootstrapping	44
3.3.2. Portierung über Zwischensprachen	45
3.3.2.1. Das UNCOL-Konzept	47
3.3.2.2. Maschinen- und sprachspezifische Zwischensprachen	49
3.3.2.3. Niveau und (Rest-) Maschinenabhängigkeit von Zwischensprachen	51
3.3.3. Ein Vergleich moderner Zwischensprachen	52
3.4. Fazit	54

4. Eine Erweiterung der Programmiersprache PASCAL um das Sprachkonstrukt Modul	56
4.1. Das Sprachkonzept Modul	57
4.1.1. Zur Syntax und Semantik von Moduln	57
4.1.2. Separate Compilierung von Moduln	64
4.2. Das Modul-Konzept im Rahmen des CGS POCO	65
5. Das CGS POCO: Die Realisierung	67
5.1. Struktur und Funktionsweise von POCO	67
5.2. Die Generator-Eingabe-Sprache(GES)	71
5.3. Die Komponenten des CGS	73
5.3.1. Der Grammatik-Leser	73
5.3.2. Der Scanner-Generator	78
5.3.3. Der Parser-Generator	80
5.3.3.1. Die Generierung einer Compile-Zeit-Fehlerbehandlung	81
5.3.3.2. Struktur der generierten Parser	83
5.3.4. Der Generator der Compile-Zeit-Attributberechnung	84
5.3.4.1. Die Realisierung der Grammatik-Transformation .	85
5.3.4.2. Die Berechnung der Attributkeller-Operationen .	86
5.3.4.3. Struktur des generierten Attribut-Behandlungs-Moduls	90
5.4. Die Struktur der generierten Compiler	91
5.5. Zusammenfassung	94
6. Die gemeinsame Portierungsschnittstelle für das Generator-System und die generierten Compiler	96
6.1. Die Portierungsaufgabe	96
6.2. Das Portierungsverfahren	97
6.2.1. Die p-Code-Stack-Maschine	99
6.2.2. Rest-Maschinenabhängigkeiten in p-Code	106
6.2.3. Zwei Beispiele für die Verwendung von p-Code	108
6.3. Die Eignung von p-Code im Rahmen der POCO-Konzeption	110
6.4. Das Portierungsverfahren für das System und die generierten Compiler	112
6.4.1. Der Portierungsaufwand auf der Generierungsmaschine	113
6.4.1.1. Behandlung von Zeichensatz und typabhängiger Speicherplatzzuordnung	114
6.4.1.2. Das Binden der separat compilierten Moduln	116

6.4.1.2.1.	Zur Code-Generierung für Moduln	117
6.4.1.2.2.	Die Funktionsweise des maschinenunabhängigen Binders	118
6.4.1.3.	Zusammenfassung des Portierungsaufwands auf der Generierungsmaschine	123
6.4.2.	Der Portierungsaufwand auf der Zielmaschine	124
6.4.3.	Eine Schätzung des Portierungsaufwands	125
7.	Direkte Generierung der Standard-Compiler-Moduln in Form von p-Code-Programmen	127
7.1.	Direkte Generierung der Parse-Tabellen in Form von p-Code-Segmenten	130
7.1.1.	Unmittelbare Umsetzung der ACTION-Tabelle in ein programmstrukturiertes p-Code-Segment	132
7.1.1.1.	Direkte Generierung von Zuweisungen der Information der Analyse-Matrix	136
7.1.1.2.	Geeignete Schemata für den Zugriff auf Analyse-Matrix-Positionen	139
7.1.1.3.	Spezielle Code-Schemata für das innere case-Statement	142
7.1.2.	Erzeugung der ACTION-Tafel als programmstrukturiertes p-Code-Segment unter Zuhilfenahme einer internen Tabellenumformung ...	148
7.1.2.1.	Generierung von Code mit Hilfe der Tabellenstruktur	151
7.1.3.	Tabellenstrukturierte Umsetzung der ACTION-Tabelle in p-Code	159
7.1.3.1.	Eine einfache Erweiterung von p-Code zur Ermöglichung der Ablage konstanter Tabellen im Konstantenspeicher	159
7.1.3.2.	Ablage generierter Tabellen im Datenspeicher der p-Maschine	162
7.1.3.2.1.	Initialisierung der Tabellen durch Feldzugriffe	163
7.1.3.2.2.	Ablage der Tabellen im Datenspeicher unter Ausnutzung der p-Maschinen-Stack-Architektur .	164
7.1.4.	Ein Vergleich der Ergebnisse bei direkter Generierung von p-Code gemäß den beschriebenen Methoden	166
7.2.	Direkte Generierung des Attribut-Übergabe- Moduls als p-Code-Segment	168
7.2.1.	Typabhängige Vergabe von Attributkeller- Adressen bei direkter p-Code-Generierung	169
7.2.2.	Die direkte Erzeugung der charakteristischen AK-Operationen in p-Code	170
8.	Zusammenfassung und Ausblick	177

Anhang

A: p-Code-Instruktionssatz in alphabetischer Ordnung	181
B: Syntax der POCO-Eingabesprache	183
C: Syntax der POCO-Eingabesprache (Syntax-Diagramme)	198
D: Beispiel für eine Eingabe in der Generator-Eingabesprache ...	202
E: Liste der POCO-Fehlermeldungen	208
F: ASCII-Zeichensatz	212
G: Hinweise zur Bedienung des Generators	213
Literaturverzeichnis	220