Inhalt

1 Einleitung 1

2 Bisherige Ansätze zur Programmentwicklung 5

2.1 Der Software-Engineering-Ansatz 5
2.2 Programmiermethodik mit formaler Grundlage 8
2.3 Deduktive Ansätze 9
2.3.1 Deduktive Programmsynthese nach Manna und Waldinger 10
2.3.2 Syntaxgesteuerte, semantikunterstützte Programmsynthese 11
2.3.3 Automatische Synthese von Skolemfunktionen 13
2.3.4 Programmsynthese mit intuitionistischer Typentheorie 14
2.4 Transformationelle Ansätze 16
2.4.1 Transformation von rekursiven Programmen 16
2.4.2 CIP 17
2.4.3 Programmsynthese mit Termersetzungssystemen 18
2.4.4 Der Bird/Meertens-Formalismus 19
2.4.5 Ein informeller Ansatz 20
2.5 Ein problemorientierter Ansatz 22
2.6 Der mit der vorliegenden Arbeit verfolgte Ansatz 23

3 Das KIV-System als Werkzeug für die formale Programmentwicklung 25

3.1 Die KIV-Logik 26
3.1.1 Syntax 26
3.1.2 Semantik 33
3.1.3 Ein Sequenzenkalkül für die dynamische Logik 39
3.2 Die Metasprache PPL 48
3.2.1 Erzeugung von Beweisbäumen 49
3.2.2 Kontrollstrukturen von PPL 52
3.3 Implementierung von Beweismethoden mit dem KIV-System 54

4 Ein programmiersprachenorientierter Ansatz 56

4.1 Syntax und Semantik von Guarded-Command-Programmen 57
4.2 Beschreibung der Heuristiken 60
4.2.1 Sätze und Strategien für die Entwicklung von bedingten Anweisungen und Schleifen 60
4.2.2 Entwicklung von Schleifeninvarianten 62
4.3 Behandlung des Indeterminismus 64
4.3.1 Änderung der Formelsemantik 65
4.3.2 Simulation mit Orakeln 66
4.3.3 Reihenfolgenunabhängige Entwicklung von bedingten Anweisungen 70
4.4 Die Rolle von Metavariablen bei der Top-Down-Programmentwicklung 71
4.5 Die implementierte Strategie 74
4.5.1 Entwicklung von zusammengesetzten Anweisungen 76
4.5.2 Entwicklung von bedingten Anweisungen 77
4.5.3 Entwicklung von Schleifen 80
4.5.4 Entwicklung von Schleifeninvarianten 89
4.6 Zusammenhang mit Guarded Commands 90
4.7 Ein Beispiel 97
4.8 Abschließende Bemerkungen 103

5 Formale Programmentwicklung durch sukzessive Etablierung von Teilzielen und Rückwärts-Schleifenentwicklung 105

5.1 Die generelle Methode 106
5.2 Problembeschreibungen, Normalform von Spezifikationen 108
5.3 Die Grundstrategie 110
5.4 Die Strengthening-Strategie 110
5.5 Die Disjoint-Goal-Strategie 114
5.6 Die Protection-Strategie 117
5.7 Die Forward-Loop-Strategie 119
5.8 Rückwärtsentwicklung von Schleifen 121
5.8.1 Das Konzept des invarianten Zieles 122
5.8.2 Die Preservation-Strategie 130
5.8.3 Die Preservation-Composition-Strategie 132

5.8.4 Die Backward-Loop-Strategie 134
5.9 Die Conditional-Strategie 135
5.10 Die Skip-Strategie 140
5.11 Die Assignment-Strategie 141
5.12 Berechnung von Nachbedingungen 142
5.13 Einführung neuer Sicherungsvariablen 144
5.14 Abschließende Bemerkungen 146

6 Vollautomatische Programmsynthese mit Finite Differencing 148

6.1 Ausgangspunkt: Ein Beispiel 148
6.2 Abstraktion des Beispiels zu einer Programmentwicklungsmethode 150
6.3 Realisierung des Verfahrens in der KIV-Umgebung 153

7 Halbautomatische Entwicklung von Divide-and-Conquer-Algorithmen 158

7.1 Die Divide-and-Conquer-Taktik 158
7.2 Designstrategien für Divide-and-Conquer-Algorithmen 162
7.3 Realisierung der Designstrategien 163
7.3.1 Bibliotheken mit Standardalgorithmen 164
7.3.2 Abgeleitete Antezedenten 166
7.3.3 Realisierung von DS1 167
7.3.4 Realisierung von DS2 168

8 Ein allgemeines Konzept zur formalen Modellierung von Top-Down-Programmentwicklungsmethoden 170

8.1 Programmierprobleme 172
8.1.1 Platzhalter 172
8.1.2 Berechnete Nachbedingungen 173
8.1.3 Definition von Programmierproblemen und ihren Lösungen 174

8.2 Programmentwicklungsstrategien 177
8.3 Ein uniformer Ablaufmechanismus 182

9 Definition einer offenen, integrierten Programmentwicklungsmethode 184

9.1 Umsetzung der Konzepte auf dynamische Logik und PPL 185
9.2 Modellierung der Methode zur sukzessiven Etablierung von Teilzielen 187
9.3 Modellierung des programmiersprachenorientierten Ansatzes 188
9.3.1 Definition von *comp-strat* 189
9.3.2 Definition von *if-strat* 190
9.3.3 Definition von *while-strat* 192
9.4 Modellierung des Finite-Differencing-Ansatzes 193
9.5 Modellierung der Divide-and-Conquer-Methode 196
9.6 Weiterentwicklung der Methode 201
9.6.1 Eine disjunktive Konditionalstrategie 201
9.6.2 Verallgemeinerung der Zuweisungsstrategie 204
9.6.3 Überführen von Implikationen in Gleichungen 206
9.7 Wiederverwendbarkeit von entwickelten Programmen 208
9.8 Entwicklung von Prozeduren 212
9.8.1 Schematische Prozedurrümpfe 212
9.8.2 Nichtschematische Prozedurrümpfe 217
9.9 Benutzung der Methode 221

10 Fazit 225

11 Literatur 227

Anhang A1 Verwendete Regeln 231

Anhang A2 Ableitungen der Axiome für Diamond-Formeln 246

Anhang A3 Taktiken und Programme des programmiersprachenorientierten Ansatzes **258**

Anhang A4 Entwicklung eines Divide-and-Conquer-Algorithmus **263**