

Inhaltsverzeichnis

1	Einleitung	1
1.1	Metaprogrammierung	2
1.1.1	Programm und Daten	3
1.1.2	Metaprogramme: Programme als Daten	5
1.1.3	Interpreter und Compiler	6
1.2	Repräsentation und Interpretation	7
1.2.1	Re-Präsentation und Auslegung	8
1.2.2	Ein Trade-Off bei komplexen Systemen	13
1.2.3	Eine Umgehung des Trade-Offs	16
1.2.4	Erweiterbare Systeme	18
1.3	Logische Programme und Prolog	20
1.3.1	Logische Programme	21
1.3.2	Deklarative Bedeutung von Horn-Klausel-Programmen ..	22
1.3.3	Prozedurale Bedeutung von Horn-Klausel-Programmen .	23
1.3.4	Prolog-Programme	24
1.3.5	Prolog-Implementationen	27
1.3.6	Ein einfaches Prolog-Programm	28
1.4	Prolog zur Metaprogrammierung	29
1.4.1	Prolog in Prolog	30
1.4.2	Der Inhalt der weiteren Kapitel dieses Buches	33

2	Ein einführendes Beispiel	35
2.1	Beispiel für eine komplexer werdende Repräsentation	35
2.2	Reduktion der Komplexität durch Einführung einer Interpretationsschicht	38
2.3	Systematisches Hinzufügen von Funktionalität zu einem Programmsystem	39
2.4	Automatische Elimination einer Interpretationsschicht für das konkrete Beispiel	42
2.5	Automatisches Hinzufügen von Funktionalität zu verschiedenen Programmen	45
3	Der Übersetzergenerator \mathcal{T}	51
3.1	Interpretative und compilative Exekution	51
3.1.1	Äquivalenz des Berechnungsverhaltens von interpretativer und compilativer Exekution	52
3.1.2	Gemeinsamkeiten von interpretativer und compilativer Exekution	54
3.1.3	Analyse- und Exekutionsschritte in Metainterpretern	58
3.2	Überführung von Interpretern in Compiler	63
3.2.1	Übersetzung von Zielen G nach G'	63
3.2.2	Übersetzung von Klauseln P nach P'	64
3.2.3	Behandlung von internen Argumenten der Interpreter	65
3.2.4	Automatische Überführung von Interpretern in Compiler	67
3.2.5	Der Aufbau von Klauseln in P'	70
3.2.6	Eine Implementation des Übersetzergenerators \mathcal{T}	71
3.2.7	Syntaktische Vereinfachung für den Aufruf kompilierter Programme	78
3.3	Laufzeitverbesserungen und algorithmische Komplexität	80
3.3.1	Meßbare Laufzeitverbesserungen	81
3.3.2	Ursprung der Laufzeitverbesserung	82
3.3.3	Algorithmische Komplexität von interpretativer und compilativer Exekution	86

4	Beispielmetainterpreter und die daraus abgeleiteten Übersetzer	91
4.1	Exkurs: Zwei Optimierungsroutinen	93
4.2	Ein einfacher, rückwärtsverkettender Interpreter	96
4.3	Ein Interpreter, der Erklärungen generiert	107
4.4	Unscharfes Schließen	108
4.5	Ein Metainterpreter mit beschränkter Deduktionstiefe	114
4.6	Ein Interpreter für Programme mit LOGO-ähnlichen Turtle-Befehlen	119
4.7	Compilation arithmetischer Ausdrücke in Prolog-Programmen ..	129
4.8	Ein Frame-Interpreter	135
4.9	Ein akademischer Interpreter	141
4.10	Ein einfacher, vorwärtsverkettender Interpreter	151
4.11	Ein Interpreter, der Ziele aufschiebt	155
4.12	Ein T-Prolog-ähnlicher Interpreter	166
4.13	Ein Metainterpreter zur Integration einer Benutzerschnittstelle ..	179
4.14	Elimination von Interpretationsebenen oder Spezialisierung von Programmen	184
4.15	Beschränkungen der Implementation	190
5	Geschachtelte Metainterpreter	197
5.1	Erklärungen aus dem Lauf des unscharf schließenden Interpreters	198
5.2	Auflösung geschachtelter Interpreter von innen nach außen	205
5.3	Erklärungen aus dem Programm zur Modellierung einer Teilerarchie	207
5.4	Laufzeitersparnisse	209
6	Einordnung und Abgrenzung zu verwandten Arbeiten	211
6.1	Partielle Evaluation	212
6.2	Ershovs „Mixed Computation“	213

6.3	Mazaher & Berry: Ableitung eines Compilers aus einer operationalen Semantik in VDL	226
6.4	Kahn & Carlsson: Compilation von Prolog-Programmen ohne Verwendung eines Compilers	227
6.5	Jones & Sestoft & Søndergaard: Die Generierung eines Compilergenerators	228
6.6	Gallagher: Transformation von logischen Programmen durch Spezialisierung von Interpretern	229
6.7	Safra & Shapiro: Metainterpreter für „wirkliche“ Anwendungen .	230
6.8	Weitere Transformationstechniken für Prolog-Programme	231
6.9	Bowens Ansatz zur Metaprogrammierung.....	231
6.10	Hoggers Programmtransformationen	232
6.11	Henschen & Naqvi: Regel-Compilation für Datalog-Sprachen ...	236
6.12	Übersetzung von Prolog-Programmen in abstrakte Maschinencodes oder in konventionelle Programmiersprachen	236
7	Schlußbemerkungen	239
A	Eine Implementation des Übersetzergenerators \mathcal{T}	243
B	Symboltabelle	249
C	Literaturverzeichnis	251
D	Index	259