

# Lecture Notes in Computer Science

Edited by G. Goos and J. Hartmanis

141

---

Uwe Kastens  
Brigitte Hutt  
Erich Zimmermann

GAG: A Practical  
Compiler Generator

---



Springer-Verlag  
Berlin Heidelberg New York 1982

**Editorial Board**

D. Barstow W. Brauer P. Brinch Hansen D. Gries D. Luckham  
C. Moler A. Pnueli G. Seegmüller J. Stoer N. Wirth

**Authors**

Uwe Kastens  
Brigitte Hutt  
Erich Zimmermann  
Institut für Informatik II der Universität Karlsruhe  
Postfach 6380, 7500 Karlsruhe 1

CR Subject Classifications (1981): D 2.1, D 3.1, D 3.2, D 3.4, F 4.2

ISBN 3-540-11591-9 Springer-Verlag Berlin Heidelberg New York  
ISBN 0-387-11591-9 Springer-Verlag New York Heidelberg Berlin

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically those of translation, reprinting, re-use of illustrations, broadcasting, reproduction by photocopying machine or similar means, and storage in data banks. Under § 54 of the German Copyright Law where copies are made for other than private use, a fee is payable to "Verwertungsgesellschaft Wort", Munich.

© by Springer-Verlag Berlin Heidelberg 1982  
Printed in Germany

Printing and binding: Beltz Offsetdruck, Hemsbach/Bergstr.  
2145/3140-543210

CONTENTS:

Chapter 1: Introduction	1
Chapter 2: The Compiler Generator GAG	5
2.1 Introduction	5
2.2 Attributed Grammars	5
2.3 System Overview	7
2.4 The Generated Compiler	10
2.4.1 Parser Interface	10
2.4.2 The Structure Tree	10
2.4.3 Attribute Evaluation	11
2.4.4 Attribute Types	11
2.4.5 I/O of the Generated Compiler	12
2.5 Dependency Analysis	12
2.6 Attribute Optimization	14
2.7 Compiler Generation	14
2.8 Experience	15
2.8.1 The Processing of Attributed Grammars	15
2.8.2 Generated Compiler Front-ends	16
Chapter 3: ALADIN - A Language for Attributed Definitions	19
3.1 Introduction	19
3.1.1 Notation	20
3.1.2 Basic Symbols	21
3.2 General Structure	22
3.3 Constant Definitions	22
3.4 Type Definitions	22
3.4.1 Enumeration Types	23
3.4.2 Subrange Types	23
3.4.3 Union Types	23
3.4.4 Structure Types	23
3.4.5 Set Types	24
3.4.6 List Types	24
3.5 Symbol and Attribute Definitions	24
3.6 Productions	25
3.7 Semantic Rules	26
3.7.1 Attribute Rules	26
3.7.2 Attribute Names	26
3.7.3 Attribute Transfer	27
3.7.4 Context Conditions	28
3.8 Semantic Expressions	28
3.8.1 Formulas	28
3.8.2 Type and Symbol Tests	29
3.8.3 Remote Attribute Access	30
3.8.4 Let Clauses	31
3.8.5 Operands	31
3.9 Semantic Clauses	32
3.9.1 Type Conversion	32
3.9.2 Calls	33
3.9.3 Case Clauses	33
3.9.4 Conditional Clauses	34
3.9.5 Relational Clauses	34
3.10 Function Definitions	34
3.10.1 Generic Functions	35
3.11 Standard Definitions	35

## IV

Chapter 4: Development of an Attributed Grammar for a Pascal-Analyzer	37
4.1 Introduction	37
4.2 Development of the Attributed Grammar	38
4.2.1 Types	39
4.2.2 Scope Rules	43
4.3 Error Handling	46
4.4 Performance and Optimizations	48
4.5 Results of Attribute Evaluation	50
Chapter 5: Generating Efficient Compiler Front-ends	53
5.1 Introduction	53
5.2 Basic Generation Concepts	54
5.2.1 Types in the Generated Compiler	55
5.2.2 The Program Tree Structure	55
5.2.3 Sequence Control of Attribute Evaluators	56
5.3 Improvement of the Tree Representation	58
5.3.1 Tree Compactification	59
5.3.2 Tree Partitioning	60
5.3.3 Attribute Evaluation Without Tree	60
5.4 Attribute Optimization	61
5.5 General Optimization Techniques	64
5.5.1 Common Subexpressions	64
5.5.2 Recursion Elimination	65
5.5.3 Inline Code for Functions	65
5.6 AG Transformations	65
5.7 Application to Pascal	66
5.8 Results	71
Appendix A: Attributed Grammar for Pascal	73
Appendix B: Results of the Usage of GAG	137
References	155