

# Lecture Notes in Computer Science

Edited by G. Goos and J. Hartmanis

46

---

F. L. Bauer, P. Brinch-Hansen, E. W. Dijkstra,  
A. Ershov, D. Gries, M. Griffiths, C. A. R. Hoare,  
G. Seegmüller, W. A. Wulf

## Language Hierarchies and Interfaces

International Summer School

Edited by F. L. Bauer and K. Samelson

---



Springer-Verlag  
Berlin · Heidelberg · New York 1976

## Editorial Board

P. Brinch Hansen · D. Gries · C. Moler · G. Seegmüller · J. Stoer  
N. Wirth

## Editors

Prof. Dr. Dr. h. c. Dr. Friedrich L. Bauer  
Prof. Dr. Klaus Samelson  
Institut für Informatik  
der Technischen Universität  
Arcisstraße 21  
8000 München 2/BRD

**Library of Congress Cataloging in Publication Data**  
Main entry under title:

Language hierarchies and interfaces.

(Lecture notes in computer science ; 46)  
"The international summer school took place from  
July 23 to August 2, 1975, in Marktoberdorf ... and was  
sponsored by the NATO Scientific Affairs Division under  
the 1975 Advanced Study Institutes programme."  
Includes bibliographical references and index.  
1. Electronic digital computers--Programming--Con-  
gresses. 2. Programming languages (Electronic computers)  
--Congresses. I. Bauer, Friedrich Ludwig, 1924-  
II. Samelson, Klaus, 1918- III. North Atlantic  
Treaty Organization. Division of Scientific Affairs.  
IV. Series.  
QA76.6.I335 001.6'42 76-54339

---

**AMS Subject Classifications (1970):** 68-02, 68A05

**CR Subject Classifications (1974):** 4.12, 4.20, 4.22, 4.30, 4.31, 4.32, 4.34,  
5.24

---

**ISBN 3-540-07994-7** Springer-Verlag Berlin · Heidelberg · New York  
**ISBN 0-387-07994-7** 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, re-  
printing, 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 the publisher, the amount of the fee to  
be determined by agreement with the publisher.

© by Springer-Verlag Berlin · Heidelberg 1976

Printed in Germany

Printing and binding: Beltz Offsetdruck, Hemsbach/Bergstr.

## P R E F A C E

The International Summer School took place from July 23 to August 2, 1975, in Markt-  
oberdorf. This Summer School was organized under the auspices of the Technical Uni-  
versity Munich, and was sponsored by the NATO Scientific Affairs Division under the  
1975 Advanced Study Institutes Programme. Partial support for this conference was  
provided by the European Research Office and the National Science Foundation.

# Contents

## INTRODUCTION

E. W. Dijkstra	ON THE TEACHING OF PROGRAMMING, I.E. ON THE TEACHING OF THINKING	1
----------------	---	---

## CHAPTER 1.: CONCURRENCY

C.A.R. Hoare	PARALLEL PROGRAMMING: AN AXIOMATIC APPROACH	11
	1. <i>Introduction</i>	12
	2. <i>Concepts and Notations</i>	12
	3. <i>Disjoint Processes</i>	14
	4. <i>Competing Processes</i>	17
	5. <i>Cooperating Processes</i>	20
	6. <i>Communicating Programs</i>	24
	7. <i>Colluding Processes</i>	29
	8. <i>Machine Traps</i>	34
	9. <i>Conclusion</i>	36
	<i>References</i>	38
	<i>Appendix</i>	40
E. W. Dijkstra	ON-THE-FLY GARBAGE COLLECTION: AN EXERCISE IN COOPERATION	43
	<i>Introduction</i>	44
	<i>Preliminary Investigations</i>	46
	<i>A Coarse-grained Solution</i>	48
	<i>A Solution with a Fine-grained Collector</i>	52
	<i>A Solution with a Fine-grained Mutator as well</i>	53
	<i>In Retrospect</i>	54
	<i>History and Acknowledgements</i>	55
	<i>References</i>	55
	<i>Appendix</i>	55
D. Gries	AN EXERCISE IN PROVING PARALLEL PROGRAMS CORRECT	57
	1. <i>Introduction</i>	58
	2. <i>Definition and Use of the Language</i>	59
	3. <i>On-the-fly Garbage Collection</i>	63
	4. <i>Proof of Correctness of the Mutator-Collector System</i>	69
	4.1. <i>Proof Outline for the Main Program</i>	71
	4.2. <i>Proof Outline for the Marking Phase</i>	71
	4.3. <i>Proof Outline for the Collecting Phase</i>	73
	4.4. <i>Proof of Properties of the Mutator</i>	73

4.5. <i>Showing Non-interference</i>	75
5. <i>Concluding Remarks</i>	78
<i>References</i>	81
P. Brinch Hansen	THE PROGRAMMING LANGUAGE CONCURRENT PASCAL 82
1. <i>The Purpose of Concurrent Pascal</i>	84
1.1. <i>Background</i>	84
1.2. <i>Processes</i>	84
1.3. <i>Monitors</i>	85
1.4. <i>System Design</i>	88
1.5. <i>Scope Rules</i>	93
1.6. <i>Final Remarks</i>	95
2. <i>The Use of Concurrent Pascal</i>	96
2.1. <i>Introduction</i>	96
2.2. <i>Processes</i>	96
2.3. <i>Monitors</i>	100
2.4. <i>Queues</i>	103
2.5. <i>Classes</i>	104
2.6. <i>Input/Output</i>	105
2.7. <i>Multiprocess Scheduling</i>	106
2.8. <i>Initial Process</i>	108
<i>Acknowledgements</i>	110
<i>References</i>	110
	 <u>CHAPTER 2.: PROGRAM DEVELOPMENT</u>
E. W. Dijkstra	GUARDED COMMANDS, NON-DETERMINACY AND A CALCULUS FOR THE DERIVATION OF PROGRAMS 111
1. <i>Introduction</i>	111
2. <i>Two Statements made from Guarded Commands</i>	112
3. <i>Formal Definition of the Semantics</i>	114
3.1. <i>Notational Prelude</i>	114
3.2. <i>The Alternative Construct</i>	116
3.3. <i>The Repetitive Construct</i>	118
4. <i>Formal Derivation of Programs</i>	119
5. <i>Concluding Remarks</i>	122
<i>Acknowledgements</i>	123
<i>References</i>	124
M. Griffiths	PROGRAM PRODUCTION BY SUCCESSIVE TRANSFORMATION 125
1. <i>Introduction</i>	126
2. <i>Successive Transformation</i>	127
2.1. <i>The Problem</i>	127
2.2. <i>Solution by Invariants</i>	128
2.3. <i>Solution by Successive Transformation</i>	129
2.4. <i>Discussion</i>	131
3. <i>Transformation Methods</i>	133
3.1. <i>Recursion and Iteration</i>	133
3.2. <i>Introduction of a Variable</i>	134
3.3. <i>Function Inversion and Counting</i>	136
3.4. <i>Changes in Data Structure</i>	138
3.5. <i>Program Schemes and Automatic Transformation</i>	139

## VII

	3.6. Discussion	140
	4. Some Implications	142
	4.1. Language Design	142
	4.2. System Structure	143
	4.3. The Multi-Language Problem	144
	4.4. Efficiency	145
	4.5. Use of Static Information	146
	5. Conclusion	147
	5.1. Associated Research	147
	5.2. Final Remarks	148
	References	149
	Acknowledgements	152
 F. L. Bauer	 PROGRAMMING AS AN EVOLUTIONARY PROCESS	 153
	First Lecture: METAMORPHOSES	155
	Styles of Programming	155
	Properties Defining Recursion and their Derivation	158
	Second Lecture: TECHNIQUES	162
	Transition between Recursion and Iterative Notation	162
	The COOPER Transformation as an Example for the Recursion Removal	165
	Function Inversion	167
	Third Lecture: DANGEROUS CORNERS	172
	Sequentialisation and the Danger of Destruction	172
	Sharing of Variables	174
	The Method of Invariants	176
	Conclusion: PROGRAMMING AS A PROCESS	179
	References	181
 C. A. R. Hoare	 PROOF OF CORRECTNESS OF DATA REPRESENTATION	 183
	1. Introduction	183
	2. Concepts and Notations	183
	3. Example	184
	4. Semantics and Implementation	186
	5. Criterion of Correctness	186
	6. Proof Method	187
	7. Proof of Smallintset	188
	7.1. Initialisation	188
	7.2. Has	188
	7.3. Insert	189
	7.4. Remove	189
	8. Formalities	190
	9. Extensions	191
	9.1. Class Parameters	191
	9.2. Dynamic Object Generation	192
	9.3. Remote Identification	192
	9.4. Class Concatenation	192
	9.5. Recursive Class Declaration	192
	References	193
 F. L. Bauer	 APPENDIX: A PHILOSOPHY OF PROGRAMMING	 194
	First Lecture:	
	A Unified, Conceptual, Basis of Programming	196

<i>Second Lecture:</i>	
<i>The Role of Structuring in Programming</i>	204
<i>Third Lecture:</i>	
<i>System Uniformity of Software and Hardware</i>	216
<i>Final:</i>	
<i>Our Responsibility</i>	227
<i>Literature</i>	229
<i>Appendix:</i>	
<i>Variables Considered Harmful</i>	230
<i>Procedures and their Parameters</i>	231
<i>Building Procedures from Primitives</i>	233
<i>Result Parameters</i>	235
<i>Variables</i>	237

### CHAPTER 3.: OPERATING SYSTEMS STRUCTURE

C.A.R. Hoare	THE STRUCTURE OF AN OPERATING SYSTEM	242
	1. Introduction	243
	2. A Class with Inner	244
	3. A Nested Class Declaration	246
	4. Compile Time Checking	247
	5. Multilevel Structuring	248
	6. A Third Level	249
	7. Error Control	250
	8. Accounting	252
	9. The Top Level	253
	10. Protection	254
	Conclusion	256
	Acknowledgements	256
	References	265
G. Seegmüller	LANGUAGE ASPECTS IN OPERATING SYSTEMS	266
	1. The Role of Language in Operating Systems	268
	1.1. Language und Function	268
	1.2. Language and People	269
	1.3. Language and Computing Systems	270
	1.4. Language and System Construction	274
	2. Are there Special Requirements for Systems Programming	277
	3. A Remark on Current Systems Programming Languages	279
	4. How Does the Successful Systems Programmer Survive	280
	5. Design Criteria for an Operating System Programming Language	281
	6. Language Mechanisms Assisting in the Con- struction of Structured Systems	282
	7. Example: The Language System ASTRA	285
	8. Concluding Remarks	289
	9. Acknowledgements	290
	10. Literature	290

W. A. Wulf	STRUCTURED PROGRAMMING IN THE BASIC LAYERS OF AN OPERATING SYSTEM	293
	<i>Introduction</i>	294
	<i>A Personal View of Structure, Programs, and Programming</i>	295
	<i>Comments on "Hierarchy"</i>	301
	<i>Layers of Operating Systems</i>	304
	<i>The "Basic" Layers - Some Assumptions</i>	311
	<i>A High Level Model of a Hydry-like System</i>	314
	<i>The Lowest Level of Hydra -- Machine Assumptions</i>	318
	<i>A Bottom-Up-Presentation</i>	320
	<i>Some Concluding Remarks and Caveats</i>	342
E. W. Dijkstra	A TIME-WISE HIERARCHY IMPOSED UPON THE USE OF A TWO-LEVEL STORE	345
	<i>Introduction</i>	346
	<i>The Role of the Replacement Algorithm in a Multiprogramming Environment</i>	348
	<i>About the Ideal Window Size</i>	350
	<i>About the Degree of Multiprogramming</i>	351
	<i>About the Adjustment of Window Size</i>	352
	<i>Monotonic Replacement Algorithms</i>	353
	<i>The Time-wise Hierarchy</i>	354
	<i>Efficiency and Flexibility</i>	355
	<i>Temptations to be Resisted</i>	356
	<i>Analyzing the Mismatch between Configuration and Workload</i>	357
	<i>Acknowledgements</i>	357
	 <u>CHAPTER 4.: PROGRAMMING SYSTEMS STRUCTURE</u>	
A. P. Ershov	PROBLEMS IN MANY-LANGUAGE SYSTEMS	358
	<i>Lecture 1:</i>	
	1. <i>Introduction and Preview of the BETA System</i>	361
	1.1. <i>Introduction</i>	361
	1.2. <i>Brief Overview of the System</i>	363
	1.3. <i>Plan of the Course</i>	366
	1.4. <i>Example</i>	366
	<i>Lecture 2:</i>	
	2. <i>Internal Language of the BETA System</i>	368
	2.1. <i>Design Concepts</i>	368
	2.2. <i>INTEL Program Scheme</i>	369
	2.3. <i>INTEL Objects</i>	370
	2.4. <i>INTEL Statements</i>	373
	2.5. <i>Transput</i>	376
	2.6. <i>Parallelism</i>	376
	2.7. <i>Discussion</i>	377



<i>Lecture 3:</i>	
3. <i>Decomposition and Synthesis in the BETA System</i>	380
3.1. <i>Introduction</i>	380
3.2. <i>Lexical Analysis. Executive Procedures</i>	382
3.3. <i>Syntactic Analysis and Parsing</i>	384
3.4. <i>Semantic Analysis and Synthesis</i>	384
3.5. <i>Lexical Information</i>	385
3.6. <i>Syntactic Information</i>	385
3.7. <i>Semantic Information</i>	386
3.8. <i>Information for Synthesis and Code Generation</i>	387
3.9. <i>Discussion</i>	388
<i>Lecture 4:</i>	
4. <i>Optimization and Code Generation in the BETA System</i>	391
4.1. <i>Collection of the Optimising Transformations</i>	391
4.2. <i>Analysis</i>	393
4.3. <i>Factorization</i>	394
4.4. <i>Preliminary Code Generation</i>	394
4.5. <i>Memory Allocation</i>	395
4.6. <i>The Coding of Subroutines and Procedures</i>	396
4.7. <i>Final Code Generation</i>	397
<i>Lecture 5:</i>	
5. <i>Compiler Writing Systems as a Factor in Unification and Comparison of Programming Languages</i>	398
5.1. <i>Introduction</i>	398
5.2. <i>Universal Executive Procedures for Synthesis</i>	390
5.3. <i>Criteria for Evaluation of Programming Languages</i>	402
5.4. <i>Data Types</i>	403
5.5. <i>Name Declarations</i>	405
5.6. <i>Resume of the Comparison</i>	407
5.7. <i>Conclusion</i>	407
<i>Acknowledgements</i>	409
<i>References</i>	410
<i>Index of Terms</i>	411