

Lecture Notes in Computer Science

Edited by G. Goos and J. Hartmanis

292

The Munich Project CIP

Volume II: The Program Transformation System CIP-S

By the CIP System Group:

F. L. Bauer, H. Ehler, A. Horsch, B. Möller,
H. Partsch, O. Paukner, and P. Pepper



Springer-Verlag

Berlin Heidelberg New York London Paris Tokyo

Editorial Board

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

Authors

F. L. Bauer

H. Ehler

B. Möller

Institut für Informatik der Technischen Universität München

Postfach 202420, 8000 München 2, Federal Republic of Germany

A. Horsch

O. Paukner

Klinikum rechts der Isar, Rechenzentrum Block A

Ismaninger Straße 22, 8000 München 80

Federal Republic of Germany

H. Partsch

Department of Informatics VI, Catholic University of Nijmegen

Toernooiveld 1, 6525 ED Nijmegen, The Netherlands

P. Pepper

Fachbereich 20 Informatik, Technische Universität Berlin

Franklinstraße 28/29, 1000 Berlin 10 (West)

CR Subject Classification (1987): D.1.0, D.2.1–2, D.2.4, D.2.6–7, D.2.9–10,
F.3.1, I.1.3, I.2.2–3, K.6.1, K.6.3–4

ISBN 3-540-18779-0 Springer-Verlag Berlin Heidelberg New York

ISBN 0-387-18779-0 Springer-Verlag New York Berlin Heidelberg

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, re-use of illustrations, recitation, broadcasting, reproduction on microfilms or in other ways, and storage in data banks. Duplication of this publication or parts thereof is only permitted under the provisions of the German Copyright Law of September 9, 1965, in its version of June 24, 1985, and a copyright fee must always be paid. Violations fall under the prosecution act of the German Copyright Law.

© Springer-Verlag Berlin Heidelberg 1987

Printed in Germany

Printing and binding: Druckhaus Beltz, Hemsbach/Bergstr.

2145/3140-543210

PREFACE

This book is the second of two volumes that present the main results having emerged from the project CIP - Computer-Aided, Intuition-Guided Programming - at the Technical University of Munich. The central theme of this project is program development by transformation, a methodology which is felt to become more and more important.

Whereas Volume I contains the description and formal specification of a wide spectrum language CIP-L particularly tailored to the needs of transformational programming, the present Volume II contains the description, formal specification, and transformational development of a system CIP-S, that is to assist a programmer in this methodology.

This work originated from two rather different motivations: On the one hand, it is to be seen as an attempt to gain methodical experience with non-toy, medium-size software projects and, in this way, to demonstrate the feasibility of the CIP approach as a software engineering discipline. On the other hand, the system is intended to incorporate recent ideas as well as experience with our own prototype system and other people's systems. Thus, in the very end, it is to constitute the basis for a practicable software development tool usable by other people either in gaining experience themselves or in producing software.

Part I deals with general issues such as "Why to use an implemented system to assist in transformational programming?" and "What are the interesting aspects with respect to transformation systems?". It also gives a brief summary of the running CIP prototype transformation system and an informal overview of the system to be dealt with in all subsequent parts. A short account of the global requirements and their implications for the organization of the system project is given and some aspects of an appropriate user environment conclude this part.

In Part II a calculus of program transformations (including induction) is presented as a theoretical basis for the entire transformation system project.

Part III starts with a more detailed and in particular more user-oriented informal collection of technical requirements for the transformation system. In its main part a formal, algebraic specification (including all design decisions) for the language-independent core of such a system can be found, whereas language-dependent aspects and issues of an appropriate user environment are deferred to Part VI. Part III closes with a kind of validation of the formal specification and a summary of experiences made in writing the formal specification.

Part IV takes Part III as a basis and demonstrates for selected system functions how running programs can be derived from the respective specifications by means of transformations. The main criterion for selection was the probable interest of the derivations. Therefore obvious or less interesting developments deliberately have been left out. As to the derivations selected, although actually done in very small steps by using the prototype system, particular emphasis has been laid on expressing the essential lines of thought rather than particular concrete rules. However, these rationales of design also have been supplemented with enough technical information such that an interested reader should be able to redo the detailed developments himself. The essential purpose of giving these selected developments in Part IV is to demonstrate that they can be done with an implemented transformation system. Many further developments of functions specified in Part III have been carried out with the prototype. They can be found in full detail in [Ehler et al. 87], out of which Part IV has actually been selected. Like Part III, Part IV closes with a summary of the experiences gained when doing the actual developments.

Part V is a collection of transformation rules used in Part IV. According to the philosophy of the language CIP-L used for specification and development these rules are differentiated into rules for the scheme language, rules for particular data types, and rules connected to particular computation structures.

Part VI is intended to give the main hints on how to extend the system core as specified in Part III to a running system exemplified with a sublanguage of CIP-L. In particular this part contains some more information on the language-dependent types (that have been left "open" in Part III), on converters between external and internal program representations, about the way of treating context conditions, semantic relations, and meta-predicates.

The report also contains an index of sorts, objects, and operations introduced, where the given page number refers to the defining occurrence in the specification. Cross-references within one part are given by section numbers only; references to other parts are made by prefixing the respective section numbers with the (roman) part numbers.

We would like to express our thanks to the Deutsche Forschungsgemeinschaft who has sponsored this research within the Sonderforschungsbereich 49 "Programmiertechnik" for ten years. Also, we gratefully acknowledge valuable criticism by the members of IFIP Working Group 2.1, notably by R. Bird, P. King, C.H. Lindsey, L.G.L.T. Meertens, S.A. Schuman, and, above all, M. Sintzoff. Moreover, we would like to thank H. Remus and D.E. Shough from the Santa Teresa Laboratory of IBM for their continuing support. Last, but by no means least, we gratefully acknowledge many helpful remarks by our (present or former) colleagues R. Berghammer, C. Delgado Kloos, F. Erhard, U. Hill-Samelson, R. Obermeier, H.-O. Riethmayer, G. Schmidt, R. Steinbrüggen, M. Wirsing, and H. Wössner as well as the speedy and competent assistance by M. Glashauser in doing the developments on the prototype system and in preparing the typescript.

Munich, October 1987

The CIP System Group

TABLE OF CONTENTS

<u>PART I : INTRODUCTION</u>	1
1. Transformational programming assisted by an implemented system	3
2. Issues of transformation systems	3
3. The CIP prototype system	4
4. Informal overview of CIP-S	6
5. Global requirements	6
6. Some aspects of an appropriate user environment	7
<u>PART II : THE TRANSFORMATION CALCULUS</u>	11
1. Introduction	13
2. Definition of the calculus	15
2.1. An algebraic view of programs and transformations	15
2.1.1. Signatures and terms	16
2.1.2. Formulas	16
2.2. Clauses	19
2.3. The calculus of inferences	20
2.3.1. Meta-deductions	21
2.3.2. Meta-inferences	22
2.3.3. About higher-level rules	23
2.3.4. Relationship to further proof principles	23
2.4. Derived meta-inferences	24
2.4.1. Language-independent derived meta-inferences	24
2.4.2. Language-dependent derived meta-inferences	24
2.5. The role of free variables	26
3. Representation of transformation tasks in the calculus	30
3.1. Genuine transformation steps	30
3.2. Compactification of development histories	33
3.3. Verification of applicability conditions	33
3.4. Reduction of applicability conditions (goal reduction)	34
3.5. "Claims"	35
4. Induction rules	35
4.1. Computational induction	37
4.1.1. Scott induction	37
4.1.2. Recursion induction	38
4.1.3. "Transformational" induction	38
4.1.4. Fixpoint induction	38
4.2. Structural induction	39
4.2.1. Term induction	39
4.2.2. Decomposition induction	39
5. Discussion	41

PART III : FORMAL SPECIFICATION

43

1.	Informal requirements	45
1.1.	Technical requirements	45
1.1.1.	Programs and program schemes	45
1.1.2.	Transformation rules and their application	45
1.1.3.	Verification of applicability conditions (goal reduction)	46
1.1.4.	Development of types and computational structures	46
1.1.5.	Documentation of a development	46
1.1.6.	Record of a goal reduction	47
1.1.7.	Further administrative tasks of the system	47
1.1.8.	User environment	47
1.2.	Example: A fictitious session with the system	47
2.	Formal specification of the system core	52
2.1.	Fundamental design decisions	53
2.1.1.	General design decisions	53
2.1.2.	Technical design decisions	54
2.2.	Preliminary remarks on the formal specification	56
2.2.1.	Structure of the specification	56
2.2.2.	Remarks on notation	57
2.3.	The system core	61
	EFFECT, SYSTEM-CORE	62
	GEN-COM	64
	NEUTR-COM	68
	DERIV-COM	69
	DEVTREE-COM	73
	MOVE-COM	78
	RED-COM	80
	CAT-COM	83
2.4.	The state	89
2.5.	The catalog-base	96
	CATALOG-BASE	97
	CATALOG	103
	MAP	107
	EMAP	108
	GROUP	110
2.6.	Reductions	112
2.7.	Derivations	115
	DERIVATION	117
	MTERM	122
	DEVTREE	127
	REL-COMP	133
	INFO	134
2.8.	Inferences	135
	INFERENCE	137
	CLAUSE	145
	FORMULA	147
	PRED-SYMB	149
2.9.	Terms	151
	INSTANCE	153

TERM	158
TPOS	168
LANGUAGE	169
OPERATOR	171
SORT	174
2.10. Basic types	176
SEQU	176
PRIMSET	180
SET	181
3. Validation of the specification: Example revisited	182
3.1. General remarks	182
3.2. The sample session in terms of system functions	183
3.3. Technical detailization of the system functions used in the sample session	186
4. Experiences	203
4.1. Experiences in using a formal specification	203
4.2. Technical experiences with algebraic specifications	205
<u>PART IV : FORMAL DEVELOPMENT OF SELECTED SYSTEM FUNCTIONS</u>	207
1. Preliminaries	209
1.1. Survey	209
1.2. From algebraic types to computation structures	209
1.2.1. Structures and algebras	209
1.2.2. Implementation of types by structures	210
1.2.3. The transition from two-valued to three-valued logic	211
1.2.4. Nondeterminate structures as implementations	212
1.2.5. Implementation of standard recursions	213
1.2.6. Implementation of descriptive operations	213
1.2.7. Implementation of modes with equality predicates	217
1.3. From computation structures to modules	219
1.4. Technical remarks	219
2. DEVTREE	220
2.1. Developments	220
2.2. Description of additional functions and modules	264
2.3. Specific rules	274
3. INFERENCE, CLAUSE, and FORMULA	276
3.1. Description of auxiliary functions	282
3.2. Developments of functions	285
3.3. Developments of selected theorems	333
4. TERM	349
4.1. Informal description of the match functions	349
4.2. Specification of auxiliary functions	350
4.3. Developments	352
5. Experiences	385
5.1. Methodological experiences with transformational developments	385
5.2. Experiences in using the CIP prototype system	386
5.3. Overall remarks in retrospect	387

<u>PART V : TRANSFORMATION RULES</u>	389
1. Remarks about rules	391
2. Rules for constructs of the scheme-language	393
2.1. Boolean expressions	393
2.2. Conditionals and guards	400
2.3. Applicative level	404
2.4. Pre-algorithmic constructs	405
2.5. Procedural level	410
3. Rules for data types and computation structures	413
3.1. Maps	413
3.2. Sets	414
3.3. Natural numbers	418
3.4. Conditional join operations	420
3.5. DEVTREE	430
3.6. CLAUSE	433
3.7. FORMULA	434
3.8. TERM	435
3.9. INSTANCE	442
3.10. OPERATOR	451
4. Technical transformation rules	452
4.1. Abstraction rules	452
4.2. Equalities	454
4.3. Change of recursion	455
4.4. Unfold, fold, and renaming	458
<u>PART VI : A SAMPLE INSTANTIATION OF THE SYSTEM FOR A CONCRETE LANGUAGE</u>	465
1. The basic objects and operations of the language	467
2. Abstract syntax: The type LANGUAGE	468
3. Conversion between external and internal form: Parser and unparser	472
4. Syntactic and semantic predicates	478
5. Generating the context conditions	479
6. Computing environment information	481
7. Inferences for propagating definedness information	485
8. Two sample transformation rules	486
REFERENCES	488
APPENDIX : BIBLIOGRAPHY OF THE PROJECT CIP (CONT.)	493
INDEX I : KEY NOTIONS	503
INDEX II : TYPES, SORTS, OPERATIONS	506
INDEX III : TRANSFORMATION RULES	520