

Lecture Notes in Computer Science

Edited by G. Goos and J. Hartmanis

271

Dominique Snyers
André Thayse

From Logic Design to Logic Programming

Theorem Proving Techniques and P-Functions



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

Dominique Snyers
André Thayse
Philipps Research Laboratory Brussels
Avenue Van Becelaere 2, Box 8, B-1170 Brussels, Belgium

CR Subject Classification (1987): I.2.3, B.1.2

ISBN 3-540-18217-9 Springer-Verlag Berlin Heidelberg New York
ISBN 0-387-18217-9 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

Contents

1	Introduction	1
1.1	The synthesis and computer implementation of algorithms	1
1.2	The Glushkov model or algorithmic state machine	2
1.3	The Karp and Miller model or parallel program schema	2
1.4	P-functions and the design of algorithms	3
1.5	P-functions and declarative programming	3
1.6	Abstract of chapter 2	4
1.7	Abstract of chapter 3	5
1.8	Abstract of chapter 4	5
1.9	List of symbols	6
2	Theorem proving and P-functions	8
2.1	Introduction	8
2.2	Types of computer implementations for algorithms	10
2.3	From theorem proving to P-functions	17
2.4	Laws for theorem proving and program synthesis	27
2.4.1	Propositional logic	27
2.4.2	First-order logic	30
2.4.3	Transformations acting on P-functions and algorithm implementation	33
2.4.4	Theorem proving and prime implicant extraction	35
2.5	Application: Systolic implementations based on logic programming	40
2.5.1	Introduction	40
2.5.2	Horn clause as prolog instruction	40
2.5.3	Prolog instruction as an elementary circuit	42
2.5.4	Sequential execution	42
2.5.5	Concurrent execution	43
2.5.6	Pipeline circuits	45
2.5.7	Prolog and the algorithmic state machine	45
2.5.8	Conclusion and connection with P-functions	49
2.6	Proofs	50
2.6.1	Proof of theorem 2.3	50
2.6.2	Proof of transformations (2.20, 2.21)	52
2.6.3	Proof of transformations (2.22, 2.23)	53

3	Grammars, logics and declarative programming	54
3.1	Introduction	54
3.2	Context-free grammars	55
3.3	Definite clause grammars	60
3.4	Grammars and logic	62
3.5	Prolog: an automatic theorem prover	69
3.6	Graphical formalisms for representing clauses and strategies	75
4	Grammars and Semantics	80
4.1	Introduction	80
4.2	Attribute grammars and P-functions	81
4.2.1	Attribute grammars	81
4.2.2	Attribute grammars and P-functions	85
4.3	Definite Clause Grammars	87
4.3.1	Definite clause grammars and Attribute grammars	87
4.3.2	DCG and Speech understanding	88
4.4	Definite Clause Translation Grammars'	93
4.4.1	DCTG and attribute grammars	93
4.4.2	DCTG and the text editor application	96
4.5	The semantic interpretation of natural language	101
4.5.1	Introduction	101
4.5.2	The predicate logic	103
4.5.3	The first order logic quantification	103
4.5.4	The modal logic	103
4.5.5	The lambda abstraction	105
4.5.6	The compositionality principle	106
4.6	Lexical Functional Grammars	110
4.6.1	Introduction	110
4.6.2	The c-structure	110
4.6.3	The f-structure	111
4.6.4	The semantic representation	118
	Bibliography	121