Lecture Notes in Computer Science

Edited by G. Goos and J. Hartmanis

261

Johann Christoph Freytag

Translating Relational Queries into Iterative Programs

Springer-Verlag

Berlin Heidelberg New York London Paris Tokyo

Author

Johann Christoph Freytag
IBM Almaden Research Center K55/801
650 Harry Road, San Jose, CA 95120, USA

CR Subject Classification (1987): D.1.1, H.2.4, I.2.2

ISBN 3-540-18000-1 Springer-Verlag Berlin Heidelberg New York
ISBN 0-387-18000-1 Springer-Verlag New York Berlin Heidelberg

Library of Congress Cataloging-in-Publication Data. Freytag, Johann Christoph, 1954-. Translating relational queries into iterative programs. (Lecture notes in computer science; 261) Thesis (Ph. D.)–Harvard University, 1985. Bibliography: p. 1. Relational data bases. 2. Electronic digital computers–Programming. 3. Functional programming languages. I. Title. II. Series.
QA76.9.D3F75 1987 005.75'6 87-13054
ISBN 0-387-18000-1 (U.S.)

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

Preface

This book is the published version of my thesis which I wrote as a Ph.D. student at Harvard University, Cambridge, MA, during 1985. Between 1982 and 1985, I spent three exciting years at this university developing the research ideas and results which are presented in this book. During this time, many people, directly and indirectly, contributed to the successful completion of this work.

I dedicate this book to my wife Susanne and my parents who both deserve my warmest thanks. Susannne's patience during my three years at Harvard made it possible for me to finish my dissertation successfully. In times of frustration her love and kindness have always been a source of strength and encouragement for me.

My parents' continuous care and guidance laid the foundations for this thesis. Their decision 25 years ago to escape from East Germany gave me the chance to become a student at Harvard.

At times, it seemed impossible to write this thesis. The infinite optimism of my advisor, Dr. Nathan Goodman, helped me overcome the difficulties finishing this dissertation. His trust in me and his far-sighted view guided me through my years at Harvard. He always challenged me and lead my research in new directions. His support and encouragement initiated my interest in functional programming and program transformation. I am grateful for his firm commitment to serve as my advisor for the last two years.

Prof. Philip A. Bernstein convinced me to return to Harvard as a Ph.D. student. Despite many changes and difficulties, he insured that I could continue my research at the Aiken Computation Laboratory successfully. His personality influenced my attitude as a researcher during my first year as a graduate student.

Special thanks also go to Prof. Harry R. Lewis and Prof. Mei Hsu who both invested their time in carefully reading my thesis. Their helpful comments improved the presentation of my ideas.

During the three years at Harvard I enjoyed the company of my fellow students. Doug Tygar deserves special thanks for his commitment to carefully read the first draft of my thesis despite the pressure I put on him. His critical comments improved this thesis in many ways. John Ramsdell introduced me to the mysterious world of functional programming. I am thankful for the many discussions and his patience in working with me. During the early stages of my research Dennis Shasha always had time for fruitful discussions which initiated many ideas. The discussions with Aviel Klausner clarified many thoughts, especially those in the first part of my thesis. Eleanor Sacks, Baiba Menke, and the rest of Aiken Lab's administrative staff provided the help and support which made everything much easier.

I am also grateful for a DAAD (Deutscher Akademischer Austauschdienst) fellowship which supported my studies at Harvard University for one year from 1979 to 1980. That time initiated my interest in becoming a Ph.D. student at this university.

Finally, I would like to thank the IBM Almaden Research Center and, in particular, my manager Dr. Laura Haas who provided me with the time and the resources to publish this thesis as a book.

This work was supported by the Office of Naval Research under grant ONR-N00014-83-K-0770.

Los Gatos, April 1987 Johann Christoph Freytag

Abstract

This book investigates the problem of translating set-oriented query specifications into iterative programs. The translation uses techniques of functional programming and program transformation.

The first part presents two algorithms which generate iterative programs from algebra-based query specifications. The first algorithm initially translates query specifications into recursive programs. Those are simplified by sets of transformation rules before the last step of the algorithm generates the final iterative form. The second algorithm uses a two level translation which generates iterative programs faster than the first algorithm. On the first level a small set of transformation rules performs structural simplification before the functional combination on the second level yields the final iterative form.

In the second part the same techniques are used to generate efficient programs for the evaluation of aggregate functions. One possible evaluation strategy is to sort the relation before applying the aggregate function, or better yet, to perform aggregation while sorting. Several transformation steps systematically generate these more efficient programs from separate specifications for the sort algorithm and the aggregate function.

Finally, the third part investigates the Lisp-dialect T as a possible implementation language for database systems.

Eigentlich weiß man nur,
wenn man wenig weiß;
mit dem Wissen wächst der Zweifel.

Die Wahlverwandtschaften
Johann Wolfgang Goethe (1749 - 1832)

Contents

List of Figures

List of Tables

Notation

$\mathcal{F}_{DB}$	Set of all database functions (DB functions).
$\mathcal{F}_{I}$	Set of form I DB functions.
$\mathcal{F}_{II}$	Set of form II DB functions.
$\mathcal{F}_{III}$	Set of form III DB functions.
L_{QEP}	Set of all well founded query evaluation plans (QEPs) which are composed of actions.
L_{DB}	Set of all database expressions (DB expressions) which are composed of DB functions.
L_{map}	Set of all well founded map expressions.
L_{λ}	Set of all λ-expressions.
Σ	Mapping from DB expressions to map expressions.
Ω	Mapping from map expressions to λ-expressions.
$(\mathcal{T}_P, \mathcal{R}_P)$	Transformation system for recursive functions.
$(\mathcal{T}_P, \mathcal{R}_U)$	Transformation system for union functions.
$(\mathcal{T}_{map}, \mathcal{R}_{map})$	Transformation system for map expressions.
$(\mathcal{T}_{\lambda}, \mathcal{R}_{\lambda})$	Transformation system for lambda expressions.
$B(f_1)$	function body of function f_1.

$s(f_1, f_2, i)$	Substitution of function expression calling f_2 as the ith parameter in the function expression calling f_1.
$S(f_1, f_2, i)$	Substitution of the body of function f_2 for the ith parameter into the body of function f_1.
$\mathcal{TR}_R$	Recursion based transformation algorithm.
$\mathcal{ID}$	Transformation algorithm to produce the ideal form which is a transformation step in algorithm $\mathcal{TR}_R$.
$\mathcal{RI}$	Transformation step in algorithm $\mathcal{TR}_R$ to generate iterative programs from ideal forms.
$\mathcal{UF}$	Unfolding step in transformation algorithm $\mathcal{ID}$.
$\mathcal{FL}$	Folding step in transformation algorithm $\mathcal{ID}$.
$\mathcal{SIM}$	Simplification step in transformation algorithm $\mathcal{ID}$ using transformation system $(\mathcal{T}_P, \mathcal{R}_P)$.
$\mathcal{UN}$	Union step in transformation algorithm $\mathcal{ID}$ using transformation system $(\mathcal{T}_P, \mathcal{R}_U)$.
$\mathcal{TR}_{map}$	Transformation algorithm based on map expressions and λ-expressions.
$\mathcal{T}_{map}$	Transformation step for map expressions in algorithm $\mathcal{TR}_{map}$.
$\mathcal{ML}$	Transformation step for λ-expressions in algorithm $\mathcal{TR}_{map}$.
$\mathcal{TR}_\lambda$	Simplification step for λ-expressions in algorithm $\mathcal{ML}$ using transformation system $(\mathcal{T}_\lambda, \mathcal{R}_\lambda)$.
$\mathcal{A}(t, r)$	Applying rule r exhaustively to term t.
$\mathcal{AP}(t_1, t_2)$	Applying lambda expression t_2 to lambda expression t_1.