An ACM Distinguished Dissertation
1982

Algorithmic Program Debugging

Ehud Y. Shapiro

The MIT Press

TP307
S1

8562552

Algorithmic Program Debugging

Ehud Y. Shapiro

The MIT Press
Cambridge, Massachusetts
London, England

© 1983 by Ehud Y. Shapiro and The Massachusetts Institute of Technology

All rights reserved. No part of this book may be reproduced in any form or by any means, electronic or mechanical, including photocopying, recording, or by any information storage and retrieval system, without permission in writing from the publisher.

This book was printed and bound in the United States of America.

Publisher's note: This format is intended to reduce the cost of publishing certain work in book form and to shorten the gap between editorial preparation and the final publication. Detailed editing and composition have been avoided by photographing the text of this book directly from the author's typescript or word-processor output.

This dissertation was presented to the Faculty of the Graduate School of Yale University on May 1982.

Apart from some stylistic changes and the correction of typographical errors, the only difference in content is in Sections 2.1.3 and 4.5. To these sections there has been added a description, without proof, of results obtained in [88, 91].

The thesis research was supported by the National Science Foundation, grant no. MCS8002447.

Library of Congress Cataloging in Publication Data

Shapiro, Ehud Y.
Algorithmic Program Debugging

(ACM distinguished dissertations)
Thesis (Ph.D.)—Yale University, 1982
Bibliography: p.
Includes index.
1. Debugging in computer science 2. Prolog (Computer program language) I. Title II. Series
QA76.6.S49 1983 001.64'2 82-24992
ISBN 0-262-19218-7

Algorithmic Program Debugging

ACM Distinguished Dissertations

1982

Abstraction Mechanisms and Language Design, by Paul N. Hilfinger

Formal Specification of Interactive Graphics Programming Languages, by William R. Mallgren

Algorithmic Program Debugging, by Ehud Y. Shapiro

此为试读,需要完整PDF请访问: www.ertongbook.com

8562552

Series Foreword

The Distinguished Doctoral Dissertation Series is a welcome outgrowth of ACM's annual contest, co-sponsored by The MIT Press, for the best doctoral dissertation in computer-related science and engineering.

During the judging process that is designed to select the annual winner, it has inevitably developed that some of the theses being considered in the final round are of such high quality that they also deserve publication. This book is one of those dissertations. In the judgment of the ACM selection committee and The MIT Press, it truly deserves special recognition as a Distinguished Doctoral Dissertation.

Dr. Ehud Shapiro wrote his thesis on "Algorithmic Program Debugging" while at Yale University. His thesis advisor was Dr. Dana Angluin, and the thesis was selected by Yale for submission to the 1982 competition. The Doctoral Dissertation Award committee of ACM recommended its publication because it productively combines elements of programming languages, environments, logic, and inductive inference to produce effective debugging aids. Its use of the PROLOG language provides an efficient implementation of the debugging algorithms.

Walter M. Carlson
Chairman, ACM Awards Committee

To Karl R. Popper
for his inspiring intellectual courage and clarity

and

to my parents, Shimon and Miriam
for bringing me up knowing that
the world is a great place to be

Acknowledgments

First and foremost, this thesis owes its shape and content to my advisor, Dana Angluin. She has directed my research since I came to Yale, and has had a great influence on what I have accomplished in these years, and what I know today. Her thoroughness and sincerity in doing this will be a model for me in the years to come. She also tried to teach me her high professional and personal standards; although I resisted as strongly as I could, there are still some traces of them in this thesis.

Drew McDermott escorted my first steps in Prolog. He gave me some insightful suggestions, which made me wish he had given me lots more. Just to name a few, the definition of incremental inductive inference, the idea of mechanizing oracle queries, and the methods that led to the algorithm that diagnoses finite failure, were all originated in some off-hand comments he made when he failed to avoid me in the corridor.

Alan Perlis and Mike Fischer also contributed to the final form of the thesis, especially to its introduction and conclusions. Without Alan's encouragement, I would not have had the nerve to make some of the bold claims I have made, and his flamboyant manner helped to counter-balance Dana's caution and serenity.

I got a lot of support from the people of the logic programming community. I have learned a lot from discussions with Lawrence Byrd, Ken Bowen, Alan Colmerauer, Maarten van Emden, Bob Kowalski, Frank McCabe, Fernando Pereira, and David Warren, to name a few.

David Warren's Prolog implementation was an indispensable research tool. First across the ocean, and then across the Arpanet, he reminded me that I am not the only Prolog hacker in the world, even though that's how it feels at Yale.

Donald Michie's enthusiasm about my research was an ultimate source of pleasure, and helped me feel that what I am doing may be worth while. Together with Alan Perlis, he showed me that intellectual curiosity and openness are ageless and statusless.

Bob Nix kept me informed on what is going on in the world, while I was too preoccupied with my work to pay attention to anything else. I think I have learnt more about computer science from discussions with him than from all of the courses I have taken together. I wish I could take him with me as an office mate wherever I go. He also devoted a lot of his time to reading and correcting my thesis, which compensated for some of my ignorance of English and Combinatorics.

Many people made specific technical contributions to the thesis, which I would like to acknowledge: Frank McCabe was the first to suggest that the query complexity of the diagnosis algorithms may be improved; only after this realization, did I begin to believe that they can be more than an exercise in formalizing some vague intuitions, but a real debugging tool. He also found how to diagnose Prolog programs with negation.

Ryszard Michalski insisted on understanding what I was doing, which made things clearer to me too; he offered some terminological improvements that made the formal machinery I used more intuitive and appealing.

Bob Moore debugged my debugging algorithm while I was at SRI, and refuted my naive belief that a binary search technique could be readily used in the diagnosis of nontermination.

David Plaisted suggested an improvement to the algorithm that diagnoses termination with incorrect output, which I accepted without hesitation. He then refuted his improvement, but finally we made it together to the current divide-and-query algorithm.

Last but not least, the Tools group at Yale, and in particular John Ellis, Steve Wood, and Nat Mishkin, made the use of our DEC-20 and its surrounding machinery tolerable, and even fun. I cannot imagine what my productivity as a programmer and as a writer would have been without Z, SM, and the kind help they

provided me. They have also contributed to the literary aspect of the thesis, by telling me (or, more accurately, by allowing me to pick and put) the Zen parable about the drawing of the fish.

In addition to the individuals mentioned above, I wish to thank the National Science Foundation, whose grant, numbered MCS8002447, supported me during the last two years.

Contents

此为试读,需要完整PDF请访问: www.ertongbook.com

Algorithms

Figures

Programs

Chapter 1

INTRODUCTION

1.1 The problem

It is evident that a computer can neither construct nor debug a program without being told, in one way or another, what problem the program is supposed to solve, and some constraints on how to solve it. No matter what language we use to convey this information, we are bound to make mistakes. Not because we are sloppy and undisciplined, as advocates of some program development methodologies may say, but because of a much more fundamental reason: we cannot know, at any finite point in time, all the consequences of our current assumptions. A program is a collection of assumptions, which can be arbitrarily complex; its behavior is a consequence of these assumptions; therefore we cannot, in general, anticipate all the possible behaviors of a given program. This principle manifests itself in the numerous undecidability results, that cover most interesting aspects of program behavior for any nontrivial programming system [85].

It follows from this argument that the problem of program debugging is present in any programming or specification language used to communicate with the computer, and hence should be solved at an abstract level. In this thesis we lay theoretical foundations for

此为试读,需要完整PDF请访问: www.ertongbook.com