Texts and Monographs in Computer Science

# Texts and Monographs in Computer Science

Suad Alagić
**Object-Oriented Database Programming**
1989. XV, 320 pages, 84 illus.

Suad Alagić
**Relational Database Technology**
1986. XI, 259 pages, 114 illus.

Suad Alagić and Michael A. Arbib
**The Design of Well-Structured and Correct Programs**
1978. X, 292 pages, 68 illus.

S. Thomas Alexander
**Adaptive Signal Processing: Theory and Applications**
1986. IX, 179 pages, 42 illus.

Michael A. Arbib, A.J. Kfoury, and Robert N. Moll
**A Basis for Theoretical Computer Science**
1981. VIII, 220 pages, 49 illus.

Friedrich L. Bauer and Hans Wössner
**Algorithmic Language and Program Development**
1982. XVI, 497 pages, 109 illus.

Kaare Christian
**A Guide to Modula-2**
1986. XIX, 436 pages, 46 illus.

Edsger W. Dijkstra
**Selected Writings on Computing: A Personal Perspective**
1982. XVII, 362 pages, 13 illus.

Edsger W. Dijkstra and Carel S. Scholten
**Predicate Calculus and Program Semantics**
1990. XII, 220 pages

Nissim Francez
**Fairness**
1986. XIII, 295 pages, 147 illus.

R.T. Gregory and E.V. Krishnamurthy
**Methods and Applications of Error-Free Computation**
1984. XII, 194 pages, 1 illus.

David Gries, Ed.
**Programming Methodology: A Collection of Articles by Members of IFIP WG2.3**
1978. XIV, 437 pages, 68 illus.

Edsger W. Dijkstra Carel S. Scholten

# Predicate Calculus and Program Semantics

Springer-Verlag New York Berlin Heidelberg
London Paris Tokyo Hong Kong

Edsger W. Dijkstra
University of Texas at Austin
Austin, TX 78712-1111
USA

Carel S. Scholten
Klein Paradys 4
7361 TD Beekbergen
The Netherlands

*Series Editor*

David Gries
Department of Computer Science
Cornell University
Ithaca, NY 14853
USA

Library of Congress Cataloging-in-Publication Data
Dijkstra, Edsger Wybe.
Predicate calculus and program semantics / Edsger W. Dijkstra, Carel S. Scholten.
p. cm. — (Texts and monographs in computer science)
ISBN-13: 978-1-4612-7924-2 e-ISBN-13: 978-1-4612-3228-5
DOI: 10.1007/978-1-4612-3228-5
1. Predicate calculus. 2. Programming languages (Electronic computers)—Semantics. I. Scholten, Carel S. II. Title. III. Series.
QA9.35.D55 1989
511.3—dc20 89-11540

Index prepared by Jim Farned of The Information Bank, Summerland, California.



Typeset by Associated Publishing Services, Ltd., United Kingdom.

9 8 7 6 5 4 3 2 1

# CHAPTER 0
# Preface

This booklet presents a reasonably self-contained theory of predicate transformer semantics. Predicate transformers were introduced by one of us (EWD) as a means for defining programming language semantics in a way that would directly support the systematic development of programs from their formal specifications.

They met their original goal, but as time went on and program derivation became a more and more formal activity, their informal introduction and the fact that many of their properties had never been proved became more and more unsatisfactory. And so did the original exclusion of unbounded nondeterminacy. In 1982 we started to remedy these shortcomings. This little monograph is a result of that work.

A possible —and even likely— criticism is that anyone sufficiently versed in lattice theory can easily derive all of our results himself. That criticism would be correct but somewhat beside the point. The first remark is that the average book on lattice theory is several times fatter (and probably less self-contained) than this booklet. The second remark is that the predicate transformer semantics provided only one of the reasons for going through the pains of publication.

Probably conditioned by years of formal program derivation, we approached the task of designing the theory we needed as an exercise in formal mathematics, little suspecting that we were heading for a few of the most pleasant surprises in our professional lives. After a few notational adaptations of the predicate calculus —so as to make it more geared to our manipulative needs— and the adoption of a carefully designed, strict format for our proofs, we found ourselves in possession of a tool that surpassed our wildest expectations. As we got used to it, it became an absolute delight to work with.

The first pleasant —and very encouraging!— experience was the killing of the myth that formal proofs are of necessity long, tedious, laborious, error-prone, and what-have-you. On the contrary, our proofs turned out to be short and simple to check, carried out —as they are— in straightforward manipulations from a modest repertoire.

For quite a while, each new and surprisingly effective proof was a source of delight and excitement, although it was intellectually not fully satisfactory that each of them was a well-crafted, but isolated, piece of ingenuity. We had our second very pleasant surprise with the development of heuristics from which most of the arguments emerged most smoothly (almost according to the principle "There is really only one thing one can do."). The heuristics turned the design of beautiful, formal proofs into an eminently teachable subject.

This experience opened our eyes for further virtues of presenting formal proofs in a strict format. Proofs thus become formal texts meeting precise consistency criteria. The advantage of this transition is traditionally viewed as a gain in trustworthiness: given the formal text, there need not be an argument whether the proposed proof is a proof or not since it can be checked objectively whether the text meets the criteria. (This is the aspect stressed in mechanical proof verification.) A further advantage of the transition is that it enables a meaningful comparison between alternative proofs of the same theorem, viz., by comparing the texts. The final advantage of the transition is the much greater homogeneity it introduces into the task of proof design, a task that becomes a challenge in text manipulation that is more or less independent of what the theorem was about. It is this greater homogeneity that opens the way for heuristics that are more generally applicable.

In the course of the process we profoundly changed our ways of doing mathematics, of teaching it, and of teaching how to do it. Consequently, this booklet is probably as much about our new appreciation of the mathematical activity as it is about programming language semantics. It is certainly this second aspect of the booklet that has induced us to go through the aforementioned pains of publication. It has taken us some time to muster the courage to admit this, for a revision of mathematical methodology seemed at first a rather presumptuous undertaking. As time went on, however, we were forced to conclude that the formal techniques we were trying out had never been given a fair chance, the evidence being the repeated observation that most mathematicians lack the tools needed for the skilful manipulation of logical formulae. We gave them a fair chance; the reader is invited to share our delight.

* * *

Before the reader embarks on the study of the material proper of this booklet, we would like to give him some advice on reading it.

The most important recommendation is to remember all the time that this is *not* a treatise on logic, or on the foundations of mathematics. This warning is all the more necessary since superficially it might look like one: it has logical symbols all over the place and starts with a long introduction about notation. The frequent occurrence of logical symbols has a simple, pragmatic explanation: we use them so extensively because they are so well-suited for our job. The long introduction on notation has an equally simple, pragmatic explanation: some new notation had to be introduced and, more importantly, existing notations had to be adapted to our manipulative needs. Our free use of the logical connectives before their formal introduction stresses once more that this is not a book on logic. In short, the logician is perfectly free to be taken aback by our naive refusal to make some of his cherished distinctions.

Our second recommendation to the reader is to approach this little monograph with an open mind and not to get alarmed whenever it deviates from the traditions in which he happens to have been educated. In particular, he should not equate convenience with convention. Doing arithmetic in Arabic numerals is objectively simpler —i.e., more convenient— than doing it in Roman numerals. The transition from verbal arguments appealing to "intuition" or "common sense" to calculational reasoning admits also in that area an equally objective notion of simplicity —i.e., of convenience— . We know from experience that for readers from some cultures it will be hard to accept that we leave all sorts of (philosophical or psychological) questions unanswered; the only answer we can offer is that we are from a pragmatic culture that deals with such questions by not raising them.

Our third recommendation to the reader is really a request, viz., to honour this booklet's brevity by reading it slowly. Our texts have a tendency of being misleadingly smooth and unusually compact. When, at the end, you wonder "Was this all?", we shall answer "Yes, this was all. And we hope you travelled long and far.".

* * *

The above was written a year ago, before we had started on our manuscript. It reflects our expectations. Now, 12 months and 420 handwritten pages later, we can look back on what we have actually done besides breaking in a new fountain pen. Needless to say, the major discrepancy between dream and reality has been the size: had we foreseen a 420-page manuscript, we would not have referred to a "booklet". Our only excuse is that, at the time, we had not firmly decided yet to include the material now covered in the last three chapters.

The trouble with writing a book is that it takes time and that, consequently, at the time of completion, authors are older —and perhaps wiser— than when they started. Our fate has been no exception: there are several things we would have done differently had we written them a year later. (For instance, concerns about punctuality would have been more concentrated and our treatment of the implication in Chapter 5 might have relied less heavily on the "Little Theory".) With the exception of a complete rewriting of Chapter 1, which at the end was no longer a proper introduction to the rest of the book, we have, however, abstained from any major overhauls of the manuscript. They would not have solved the problem that authors may continue to grow and that, consequently, texts have an unavoidable tendency to be more or less dated. It is in this connection that we would like to include a general disclaimer: though our enthusiasm might sometimes suggest differently, we nowhere pretend that our work leaves no room for improvement, simplification, or meaningful generalization. (We did, for instance, not explore what meaningful partial orders on programs could be introduced.)

Apart from exceeding the originally envisaged size, we have remained faithful to our original intentions, in particular to our intention of writing a monograph reflecting the current state of our art. Though we have successfully covered most of its material in graduate courses at both sides of the Atlantic Ocean, this book should not be regarded (or judged) as a textbook, because it was not intended that way. (Hence, for instance, the total absence of exercises.) There were several reasons for not wishing to write a textbook. A practical reason was that different educational systems promote totally different perceptions of student needs to be catered for in an "acceptable" textbook, and that we did not want to waste our time trying to meet all sorts of conflicting requirements. A more fundamental reason was that we think the whole notion of "a textbook tailored to student needs" certainly at graduate level much too condescending. At that stage there is nothing wrong with confronting students with material for which they have not been adequately prepared, for how else are they going to discover and to learn how to cope with the unavoidable gaps in their education? So, though we know that this book can provide the underlying material for a fascinating and highly instructive course, it is *not* a textbook and its "target audience" is just "To whom it may concern".

We have, of course, our idea about whom it concerns: the mathematically inclined computing scientists and the mathematicians with methodological and formal interests. We most sincerely hope to reach them, to thrill them, and to inspire them to improve their own work or ours. Honesty compels us to add to this wish that there is one possible —and, alas, likely— "improvement" we are not waiting for, viz., the translation of our theory into set-theoretical terminology —by interpreting predicates as characteristic

functions of subsets of states— so as to make it all more familiar. Little is so regrettable as to see one's work "improved upon" by the introduction of traditional complications one has been very careful to avoid. Hybrid arguments, partly conducted in terms of a formal system and partly conducted in terms of a specific model for that formal system, present a typical example of such confusing complications. In this connection we would like to stress that the existence of individual machine states enters the picture only when our theory is applied to program semantics, i.e., to an environment in which the individual machine state is a meanigful concept; the theory itself does not need a postulate of the existence of individual states and, therefore, should not be cluttered by their introduction.

* * *

It is a pleasure to mention here our great appreciation for our former employers, Burroughs Corporation and N.V. Philips, respectively, which loyally supported us when we embarked in 1982 on the investigations reported in this volume.

More profound gratitude than to anyone else is due to W. H. J. Feijen and A. J. M. van Gasteren who were at that time close collaborators of one of us (EWD). They did not only witness from close quarters our first formal efforts at establishing a theory of predicate transformer semantics, they can trace their contributions and their influence all through this book. Feijen is essentially the inventor of our proof format; he took the decision to write the hint between the formulae connected by it and he was the one who insisted on the discipline of allotting at least a full line to each hint. (He probably realized that this insistence was necessary for establishing a tradition in which the hints given would be as explicit as he wanted them to be.) Van Gasteren provided the rationale for this invention (and also for the invention of the square brackets to denote the "everywhere" operator): her earlier explorations had convinced us that the type of brevity thus obtained is indispensable. Later she insisted on exorcizing mathematical rabbits —pulled out of a hat— and provided the "follows from" operator as one of the means to that end.

Furthermore, we thank all colleagues, students, and members of the Tuesday Afternoon Clubs in Eindhoven and Austin, whose reactions to the material shown have helped in shaping this book. We are particularly grateful for the more detailed comments that Jayadev Misra and Lincoln A. Wallen gave on the almost final version of the manuscript; the decision to rewrite Chapter 1 has been triggered by their comments.

Finally, we express our gratitude to W. H. J. Feijen, David Gries, and Gerhard Rossbach of Springer-Verlag, New York. In their offers of assistance

in the final stages of book production, each of them has gone beyond the call of duty.

July 1988
Austin, TX, USA
Beekbergen, The Netherlands

Edsger W. Dijkstra
Carel S. Scholten

# Contents