GRADUATE TEXTS IN COMPUTER SCIENCE

Editors

David Gries

Fred B. Schneider

Springer Science+Business Media, LLC

GRADUATE TEXTS IN COMPUTER SCIENCE

Apt and Olderog, Verification of Sequential and Concurrent Programs, Second Edition

Back and von Wright, Refinement Calculus

Fitting, First-Order Logic and Automated Theorem Proving, Second Edition

Li and Vitányi, An Introduction to Kolmogorov Complexity and Its Applications, Second Edition

Munakata, Fundamentals of the New Artificial Intelligence

Nerode and Shore, Logic for Applications, Second Edition

Schneider, On Concurrent Programming

Smith, A Recursive Introduction to the Theory of Computation

Socher-Ambrosius and Johann, Deduction Systems

Ralph-Johan Back Joakim von Wright

REFINEMENT CALCULUS

A Systematic Introduction

With 27 Illustrations

Springer

Ralph-Johan Back
Joakim von Wright
Department of Computer Science
Åbo Akademi University
Lemminkäisenkatu 14
FIN-20520 Turku, Finland
backrj@abo.fi
jwright@abo.fi

Series Editors

David Gries
Fred B. Schneider

Department of Computer Science
Cornell University
Upson Hall
Ithaca, NY 14853-7501, USA

Library of Congress Cataloging-in-Publication Data
Back, Ralph-Johan.

Refinement calculus : a systematic introduction / Ralph-Johan
Back, Joakim von Wright.

p. cm. — (Graduate texts in computer science)

ISBN 978-0-387-98417-9 ISBN 978-1-4612-1674-2 (eBook)

DOI 10.1007/978-1-4612-1674-2

1. Electronic digital computers—Programming. 2. Calculus.
3. Computer logic. I. Wright, J. von (Joakim), 1955—

II. Title. III. Series: Graduate texts in computer science

(Springer-Verlag New York Inc.)

QA76.6.B28 1998

005.1'4—dc21

97-47336

Printed on acid-free paper.

© 1998 Springer Science+Business Media New York
Originally published by Springer-Verlag New York, Inc. in 1998

All rights reserved. This work may not be translated or copied in whole or in part without the written permission of the publisher (Springer Science+Business Media, LLC), except for brief excerpts in connection with reviews or scholarly analysis. Use in connection with any form of information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed is forbidden.

The use of general descriptive names, trade names, trademarks, etc., in this publication, even if the former are not especially identified, is not to be taken as a sign that such names, as understood by the Trade Marks and Merchandise Marks Act, may accordingly be used freely by anyone.

Production managed by Lesley Poliner; manufacturing supervised by Thomas King.
Photocomposed copy prepared from the authors' L^AT_EX files.

9 8 7 6 5 4 3 2 1

ISBN 978-0-387-98417-9

Preface

The refinement calculus is a framework for reasoning about correctness and refinement of programs. The primary application of the calculus is the derivation of traditional imperative programs, where programs are constructed by stepwise refinement. Each refinement step is required to preserve the correctness of the previous version of the program. We study the basic rules for this kind of program derivation in detail, looking at specification statements and their implementation, how to construct recursive and iterative program statements, and how to use procedures in program derivations. The refinement calculus is an extension of Dijkstra's weakest precondition calculus, where program statements are modeled as predicate transformers.

We extend the traditional interpretation of predicate transformers as executable program statements to that of contracts that regulate the behavior of competing agents and to games that are played between two players. The traditional interpretation of predicate transformers as programs falls out of this as a special case, where one of the two players has only a very rudimentary role to play. Our extension yields a number of new potential applications for the refinement calculus theory, like reasoning about correctness of interactive program statements and analyzing two-person games and their winning strategies.

Program refinement involves reasoning in many domains. We have to reason about properties of functions, predicates and sets, relations, and program statements described as predicate transformers. These different entities are in fact mathematically all very similar, and they form the refinement calculus hierarchy. In this book we examine the ways in which these different domains are related to each other and how one can move between them. Lattice theory forms a unifying conceptual basis

that captures similarities between the domains, while higher-order logic serves as a common logic that we can use to reason about entities in these domains.

We present a new foundation for the refinement calculus based on lattice theory and higher-order logic, together with a simple theory of program variables. These topics are covered in the first part of the book. Higher-order logic is described in a new way that is strongly influenced by lattice theory. This part also introduces notions needed for imperative programs, program variables, expressions, assignment statements, blocks with local variables, and procedures. The second part of the book describes the predicate transformer approach to programming logic and program semantics, as well as the notion of program refinement. We also study the operational semantics of contracts, games, and program statements, and show how it is related to the predicate transformer interpretation of these. The third part of the book shows how to handle recursion and iteration in the refinement calculus and also describes how to use the calculus to reason about games. It also presents case studies of program refinement. The last part of the book studies more specific issues related to program refinement, such as implementing specification statements, making refinements in context, and transforming iterative structures in a correctness-preserving way.

The refinement calculus has been a topic of investigation for nearly twenty years now, and a great number of papers have been published in this area. However, a systematic introduction to the mathematical and logical basis for program refinement has been missing. The information required to study this field is scattered through many journal and conference papers, and has never been presented as a coherent whole. This book addresses this problem by giving a systematic introduction to the field. The book partly summarizes work that has been published earlier, by ourselves and by other authors, but most of the material is new and has not been published before.

The emphasis in this book is on the mathematical and logical basis of program refinement. Although we do give examples and case studies of program construction, this is not primarily a hands-on book on how to construct programs in practice. For reasons of space we have been forced to omit a great deal of material that we wanted to include in the book. In particular, we decided not to include the important topics of data refinement and parallel and reactive system refinement here (though a simple approach to data refinement is described in Chapter 17). We plan to treat these topics in a comprehensive way in a separate book.

The book is intended for graduate students and advanced undergraduates interested in the mathematics and logic needed for program derivations, as well as for programmers and researchers interested in a deeper understanding of these issues. The book provides new insight into the methods for program derivations, as well as new understanding of program semantics and of the logic for reasoning about programs.

The book should be easy to use in teaching. We have tried to explain things quite carefully, and proofs are often given in reasonable detail. We have also included quite a number of examples and exercises. A previous understanding of logic and programming methodology is helpful but not necessary in order to understand the material in the book. The first, second, and third parts of the book can be given as successive courses, with suitable material added from the fourth part.

Acknowledgments

This book was inspired and influenced by a number of other researchers in the field. In particular, Edsger Dijkstra's work on predicate transformers and his systematic approach to constructing programs has been an important source of inspiration, as has the work of Tony Hoare on the logic of programming. Jaco de Bakker's work on programming language semantics has formed another strong influence, in particular the stringent way in which he has approached the topic. We very much appreciate Carroll Morgan's work on the refinement calculus, which has provided many ideas and a friendly competitive environment that has been much appreciated. Finally, Michael Gordon's work has shown the usefulness of higher-order logic as a framework for reasoning about both hardware and software and about formal theories in general.

Our close coworkers Reino Kurki-Suonio and Kaisa Sere deserve a special thanks for many years of fruitful and inspiring collaboration in the area of programming methodology and the logic for systematic program construction.

We are indebted to many other colleagues for numerous enlightening discussions on the topics treated in this book. Our thanks go to Sten Agerholm, Roland Backhouse, Eike Best, Michael Butler, Rutger Dijkstra, Wim Feijen, Nissim Francez, Marcel van de Goot, David Gries, Jim Grundy, John Harrison, Ian Hayes, Eric Hehner, Wim Hesselink, Peter Hofstee, Cliff Jones, Bengt Jonsson, He Jifeng, Joost Kok, Rustan Leino, Alain Martin, Tom Melham, David Nauman, Greg Nelson, John Tucker, Amir Pnueli, Xu Qiwen, Emil Sekerinski, Jan van de Snepscheut, Mark Staples, Reino Vainio, and Wolfgang Weck.

We also want to thank our students, who have provided an inspiring environment for discussing these topics and have read and given detailed comments on many previous versions of the book: Thomas Beijar, Martin Büchi, Eric Hedman, Philipp Heuberger, Linas Laibinis, Thomas Långbacka, Leonid Mikhajlov, Anna Mikhajlova, Lasse Nielsen, Rimvydas Rukšėnas, Mauno Rönkkö, and Marina Waldén.

The Department of Computer Science at Abo Akademi University and Turku Centre for Computer Science (TUCS) have provided a most stimulating working environment. The sabbaticals at the California Institute of Technology, Cambridge

University, and Utrecht University have also been very fruitful and are hereby gratefully acknowledged. The generous support of the Academy of Finland and the Ministry of Education of Finland for the research reported in this book is also much appreciated.

Last but not least, we want to thank our (sometimes more, sometimes less) patient wives and children, who have been long waiting for this book to be finished: Barbro Back, Pia, Rasmus, and Ida Back, Birgitta Snickars von Wright, Heidi, Minea, Elin, and Julius von Wright.

Ralph-Johan Back
Joakim von Wright

Department of Computer Science
Åbo Akademi University
June 30, 1997

Contents

Preface	v
1 Introduction	1
1.1 Contracts	1
1.2 Using Contracts	6
1.3 Computers as Agents	9
1.4 Algebra of Contracts	10
1.5 Programming Constructs	12
1.6 Specification Constructs	15
1.7 Correctness	18
1.8 Refinement of Programs	20
1.9 Background	21
1.10 Overview of the Book	24
I Foundations	27
2 Posets, Lattices, and Categories	29
2.1 Partially Ordered Sets	29
2.2 Lattices	36
2.3 Product and Function Spaces	40
2.4 Lattice Homomorphisms	43
2.5 Categories	49
2.6 Summary and Discussion	54
2.7 Exercises	54
3 Higher-Order Logic	57
3.1 Types and Terms	57
3.2 Semantics	62
3.3 Deductive Systems	63
3.4 Theories	65
3.5 Summary and Discussion	66
3.6 Exercises	67

4	Functions	69
4.1	Properties of Functions	69
4.2	Derivations	73
4.3	Definitions	77
4.4	Product Type	80
4.5	Summary and Discussion	82
4.6	Exercises	83
5	States and State Transformers	85
5.1	State Transformers	85
5.2	State Attributes and Program Variables	86
5.3	Reasoning with Program Variables	91
5.4	Straight-Line Programs	97
5.5	Procedures	98
5.6	Blocks and Value Parameters	101
5.7	Summary and Discussion	107
5.8	Exercises	108
6	Truth Values	109
6.1	Inference Rules for Boolean Lattices	109
6.2	Truth Values	111
6.3	Derivations with Logical Connectives	112
6.4	Quantification	115
6.5	Assumptions	118
6.6	Derivations with Local Assumptions	120
6.7	Summary and Discussion	123
6.8	Exercises	124
7	Predicates and Sets	127
7.1	Predicates and Sets	127
7.2	Images and Indexed Sets	130
7.3	Inference Rules for Complete Lattices	131
7.4	Bounded Quantification	133
7.5	Selection and Individuals	135
7.6	Summary and Discussion	136
7.7	Exercises	137
8	Boolean Expressions and Conditionals	139
8.1	Boolean Expressions	139
8.2	Reasoning About Boolean Expressions	141
8.3	Conditional Expressions	143
8.4	Proving Properties About Conditional State Transformers	146
8.5	Summary and Discussion	148
8.6	Exercises	149

9	Relations	151
9.1	Relation Spaces	151
9.2	State Relation Category	153
9.3	Coercion Operators	154
9.4	Relational Assignment	155
9.5	Relations as Programs	159
9.6	Correctness and Refinement	161
9.7	Summary	164
9.8	Exercises	164
10	Types and Data Structures	167
10.1	Postulating a New Type	167
10.2	Constructing a New Type	170
10.3	Record Types	172
10.4	Array Types	175
10.5	Dynamic Data Structures	179
10.6	Summary and Discussion	181
10.7	Exercises	181
II	Statements	185
11	Predicate Transformers	187
11.1	Satisfying Contracts	187
11.2	Predicate Transformers	189
11.3	Basic Predicate Transformers	192
11.4	Relational Updates	195
11.5	Duality	198
11.6	Preconditions and Guards	199
11.7	Summary and Discussion	200
11.8	Exercises	202
12	The Refinement Calculus Hierarchy	203
12.1	State Categories	203
12.2	Homomorphisms	206
12.3	Summary and Discussion	211
12.4	Exercises	212
13	Statements	213
13.1	Subtyping, Sublattices and Subcategories	213
13.2	Monotonic Predicate Transformers	217
13.3	Statements and Monotonic Predicate Transformers	219
13.4	Derived Statements	223
13.5	Procedures	227

13.6	Summary and Discussion	228
13.7	Exercises	230
14	Statements as Games	233
14.1	Game Interpretation	233
14.2	Winning Strategies	239
14.3	Correctness and Refinement	244
14.4	Summary and Discussion	246
14.5	Exercises	246
15	Choice Semantics	249
15.1	Forward and Backward Semantics	249
15.2	Comparing Semantics for Contract Statements	252
15.3	Refinement in the Choice Semantics	254
15.4	Summary and Discussion	255
15.5	Exercises	256
16	Subclasses of Statements	259
16.1	Homomorphic Predicate Transformers	259
16.2	Subcategories of Statements	264
16.3	Summary and Discussion	266
16.4	Exercises	267
17	Correctness and Refinement of Statements	269
17.1	Correctness	269
17.2	Stepwise Refinement	273
17.3	Refinement Laws	280
17.4	Refinement in Context	287
17.5	Refinement with Procedures	291
17.6	Example: Data Refinement	292
17.7	Summary and Conclusions	295
17.8	Exercises	297
III	Recursion and Iteration	299
18	Well-founded Sets and Ordinals	301
18.1	Well-Founded Sets and Well-Orders	301
18.2	Constructing the Natural Numbers	303
18.3	Primitive Recursion	305
18.4	Ordinals	307
18.5	Ordinal Recursion	310
18.6	How Far Can We Go?	312
18.7	Summary and Discussion	314
18.8	Exercises	314

19 Fixed Points	317
19.1 Fixed Points	317
19.2 Fixed points as Limits	320
19.3 Properties of Fixed Points	322
19.4 Summary and Discussion	326
19.5 Exercises	327
20 Recursion	329
20.1 Fixed Points and Predicate Transformers	329
20.2 Recursion Introduction and Elimination	334
20.3 Recursive Procedures	336
20.4 Example: Computing the Square Root	342
20.5 Summary and Discussion	343
20.6 Exercises	344
21 Iteration and Loops	347
21.1 Iteration	347
21.2 Properties of Iterations	350
21.3 Correctness of Iterations	352
21.4 Loops	353
21.5 Loop Correctness	356
21.6 Loop Introduction and Elimination	359
21.7 Summary and Discussion	361
21.8 Exercises	362
22 Continuity and Executable Statements	365
22.1 Limits and Continuity	365
22.2 Continuity of Statements	367
22.3 Continuity of Derived Statements	371
22.4 Executable Statements	373
22.5 Interactive Guarded Commands	377
22.6 Summary and Discussion	380
22.7 Exercises	381
23 Working with Arrays	383
23.1 Resetting an Array	383
23.2 Linear Search	386
23.3 Selection Sort	388
23.4 Counting Sort	390
23.5 Summary and Discussion	402
23.6 Exercises	402

24 The N-Queens Problem	403
24.1 Analyzing the Problem	403
24.2 First Step: The Terminating Case	406
24.3 Second Step: Extending a Partial Solution	407
24.4 Third Step: Completing for Recursion Introduction	409
24.5 Final Result	411
24.6 Summary and Discussion	411
24.7 Exercises	412
25 Loops and Two-Person Games	413
25.1 Modeling Two-Person Games	413
25.2 Winning Strategies	415
25.3 Extracting Winning Strategies	417
25.4 Strategy Improvement	421
25.5 Summary and Discussion	422
25.6 Exercises	422
IV Statement Subclasses	425
26 Statement Classes and Normal Forms	427
26.1 Universally Conjunctive Predicate Transformers	427
26.2 Conjunctive Predicate Transformers	429
26.3 Disjunctive Predicate Transformers	431
26.4 Functional Predicate Transformers	432
26.5 Continuous Predicate Transformers	435
26.6 Homomorphic Choice Semantics	437
26.7 Summary and Discussion	443
26.8 Exercises	444
27 Specification Statements	447
27.1 Specifications	448
27.2 Refining Specifications by Specifications	449
27.3 Combining Specifications	453
27.4 Refining Conjunctive Specifications	457
27.5 General Refinement of Specifications	458
27.6 Summary and Discussion	460
27.7 Exercises	461
28 Refinement in Context	463
28.1 Taking the Context into Account	463
28.2 Rules for Propagating Context Assertions	467
28.3 Propagation in Derived Statements	471
28.4 Context Assumptions	475

28.5 Summary and Discussion	477
28.6 Exercises	478
29 Iteration of Conjunctive Statements	479
29.1 Properties of Fixed Points	479
29.2 Relating the Iteration Statements	482
29.3 Iteration of Meets	484
29.4 Loop Decomposition	486
29.5 Other Loop Transformations	487
29.6 Example: Finding the Period	489
29.7 Summary and Discussion	494
29.8 Exercises	494
Appendix	497
References	501
Index	509