University of California

Los Angeles

Trustworthy Tools for Trustworthy Programs: A Mechanically Verified Verification Condition Generator for the Total Correctness of Procedures

A dissertation submitted in partial satisfaction

of the requirements for the degree

Doctor of Philosophy in Computer Science

by

Peter Vincent Homeier

1995

© Copyright by

Peter Vincent Homeier

1995

The dissertation of Peter Vincent Homeier is approved.

Rajive Bagrodia

Donald A. Martin

D. Stott Parker

David F. Martin, Committee Chair

University of California, Los Angeles

1995

I dedicate this dissertation

to the most wonderful friend I have ever had,

my Messiah, Lord, and Savior, the Son of God,

Y'shua Ha Mashiach, Jesus the Christ.

He has cared for every need faithfully,

in the midst of earthquake and opposition.

He has wholeheartedly poured out His Holy Spirit on me.

At each point of difficulty, at each resistant problem,

like a cool drop of rain, He quietly dropped the answer into me.

When the ultimate impossible cliff arose before me,

He opened doors of understanding,

drawing me beyond what I thought was the end,

through the darkness of the grave

to the dawn of a new morning.

He has been the lifter of my head, and the restorer of my hopes.

Of all the people I know,

He is the most precious to me.

He loved me enough to humbly go to the Cross and die in my place.

I can never repay such a pure and shattering gift.

This dissertation is only the smallest of ways

I can express my heart's wonder and love

for such a greater love He has lavished on me.

Table of Contents

I Background **1**

1 Introduction **3**

2 Underlying Technologies **13**

2.1 Syntax 14

2.2 Semantics 16

2.3 Partial and Total Correctness 18

2.4 Hoare Logics 20

2.5 Soundness and Completeness 22

2.6 Verification Condition Generators 24

2.7 Higher Order Logic 26

2.7.1 Higher Order Logic as a Logic 27

2.7.2 Higher Order Logic as a Mechanical Proof Assistant 29

2.8 Embeddings 31

3 Survey of Previous Research **35**

3.1 Expressions with Side Effects 36

3.2 Procedures 37

3.3 Total Correctness of Mutually Recursive Procedures 39

3.3.1 Sokołowski 40

3.3.2 Apt 41
3.3.3 America and de Boer 42
3.3.4 Pandya and Joseph 42
3.4 Verification Condition Generators, Embeddings, and Mechanically Verified Axiomatic Semantics 43
3.4.1 Ragland 45
3.4.2 Igarashi, London, and Luckham 45
3.4.3 Boyer and Moore 46
3.4.4 Gray 46
3.4.5 Gordon 47
3.4.6 Agerholm 48
3.4.7 Melham 48
3.4.8 Camilleri and Melham 48
3.4.9 Zhang, Shaw, Olsson, Levitt, *et. al.* 49
3.4.10 Lin 49
3.4.11 Kaufmann 50
3.4.12 Homeier and Martin 50

4 Organization of Dissertation 53

II Results 55

5 Sunrise 57

5.1 Programming Language Syntax 60
5.2 Informal Semantics of Programming Language 63
5.2.1 Numeric Expressions 63
5.2.2 Lists of Numeric Expressions 64
5.2.3 Boolean Expressions 64
5.2.4 Commands 65
5.2.5 Declarations 67
5.2.6 Programs 71
5.3 Assertion Language Syntax 71
5.4 Informal Semantics of Assertion Language 73
5.4.1 Numeric Expressions 74
5.4.2 Lists of Numeric Expressions 74
5.4.3 Boolean Expressions 74
5.5 Formal Semantics 76
5.5.1 Programming Language Structural Operational Semantics 78
5.5.2 Assertion Language Denotational Semantics 83
5.6 Procedure Entrance Semantic Relations 86
5.7 Termination Semantic Relations 88

6 Program Logics 91

6.1 Total Correctness of Expressions 99
6.1.1 Closure Specification 101

6.1.2 Numeric Expression Specification 102
6.1.3 Expression List Specification 103
6.1.4 Boolean Expression Specification 104
6.2 Hoare Logic for Partial Correctness 106
6.2.1 Partial Correctness Specification 106
6.2.2 Partial Correctness Rules 108
6.3 Procedure Entrance Logic 110
6.3.1 Entrance Specification 111
6.3.2 Precondition Entrance Specification 115
6.3.3 Calls Entrance Specification 116
6.3.4 Path Entrance Specification 117
6.3.5 Recursive Entrance Specification 123
6.4 Termination Logic 125
6.4.1 Command Conditional Termination Specification 126
6.4.2 Procedure Conditional Termination Specification 129
6.4.3 Command Termination Specification 129
6.5 Hoare Logic for Total Correctness 132
6.5.1 Total Correctness Specification 132

7 Verification Condition Generator 135

7.1 Definitions 136
7.1.1 Verification of Commands 136

7.1.2 Verification of Declarations 138
7.1.3 Verification of Call Graph 139
7.1.4 Verification of Programs 149
7.2 Verification Conditions 151
7.2.1 Program Structure Verification Conditions 152
7.2.2 Call Graph Structure Verification Conditions 154
7.3 VCG Soundness Theorems 161

8 Example Runs 175
8.1 Quotient/Remainder 176
8.2 McCarthy's "91" Function 182
8.3 Odd/Even Mutual Recursion 187
8.4 Pandya and Joseph's Product Procedures 201
8.5 Cycling Termination 213

9 Source Code 227

III Tour of Interesting Aspects 229

10 Partial Correctness 231
10.1 Variants 231
10.2 Substitution 234
10.2.1 Assertion Language Expression Substitution 235

10.2.2 Variables-for-Variables Substitution 239
10.2.3 Programming Language Substitution 243
10.3 Translation 247
10.4 Well-Formedness 251
10.4.1 Informal Description 253
10.4.2 Well-Formedness Predicate Definitions 256
10.5 Semantic Stages 264

11 Total Correctness 269
11.1 Reprise 271
11.1.1 Entrance Logic 271
11.1.2 Termination Logic 272
11.1.3 Recursiveness 272
11.2 Termination 273
11.2.1 Sketch of Proof 274
11.2.2 Termination of Deep Calls 276
11.2.3 Existence of an Infinite Sequence 278
11.2.4 Consequences of an Infinite Sequence 282
11.2.5 Strictly Decreasing Sequences 283

IV Conclusions 289

12 Significance 291

13 Ease of Use 297

13.1 Burden of Annotation 297

13.2 Burden of Proof 301

13.3 Areas of VCG Support 302

14 Future Research 303

14.1 Language Extensions 304

14.2 VCG Improvements 307

14.3 Implementations 307

14.4 Completeness 308

15 Conclusions 309

References 311

List of Figures

6.1 Comparison of Partial Correctness and Entrance Specifications. 96

6.2 Procedure Call Graph for Odd/Even Example. 122

7.1 Definition of *vcg*1, helper VCG function for commands. 137

7.2 Definition of *vcgc*, main VCG function for commands. 138

7.3 Definition of *vcgd*, VCG function for declarations. 139

7.4 Definition of *extend_graph_vcs* and *fan_out_graph_vcs*. 140

7.5 Definition of *graph_vcs*. 143

7.6 Definition of *vcgg*, the VCG function to analyze the call graph. . 144

7.7 Procedure Call Graph for Odd/Even Example. 144

7.8 Procedure Call Tree for Odd/Even Example. 145

7.9 Definition of *mkenv*. 149

7.10 Definition of *proc_names*. 150

7.11 Definition of *vcg*, the main VCG function. 150

7.12 Procedure Call Tree for Recursion for Odd/Even Example. . . . 156

7.13 Procedure Call Tree for Single Recursion for Odd/Even Example. 157

7.14 Diverted and Undiverted Verification Conditions for Odd/Even. 159

8.1 Procedure Call Graph for Quotient/Remainder Program. 177

8.2 Procedure Call Tree for root procedure *quotient_remainder*. . . 177

8.3 Procedure Call Graph for McCarthy's "91" Program. 183

8.4 Procedure Call Tree for root procedure *p*91. 184

8.5 Procedure Call Graph for Odd/Even Program. 189

8.6 Procedure Call Tree for root procedure *odd*. 189

8.7 Procedure Call Tree for root procedure *even*. 190

8.8 Pandya and Joseph's Product Procedures. 203

8.9 Pandya and Joseph's Proof Skeleton for procedure *product*. . . . 204

8.10 Pandya and Joseph's Proof Skeletons for procedures *oddproduct* and *evenproduct*. 205

8.11 Sunrise Proof Skeletons for procedures *product* and *oddproduct*. 207

8.12 Sunrise Proof Skeleton for procedure *evenproduct*. 208

8.13 Procedure Call Graph for Pandya and Joseph's Product Program.209

8.14 Procedure Call Tree for root procedure *product*. 210

8.15 Procedure Call Tree for root procedure *oddproduct*. 211

8.16 Procedure Call Tree for root procedure *evenproduct*. 212

8.17 Procedure Call Graph for Cycling Termination Program. 216

8.18 Procedure Call Tree for root procedure *pedal*. 216

8.19 Procedure Call Tree for root procedure *coast*. 217

List of Tables

2.1 Example programming language. 14

2.2 Example programming language structural operational semantics. 17

2.3 Floyd/Hoare Partial and Total Correctness Semantics. 19

2.4 Example programming language axiomatic semantics. 21

2.5 Example Verification Condition Generator. 25

5.1 Sunrise programming language. 59

5.2 Sunrise programming language types of phrases. 60

5.3 Sunrise programming language constructor functions. 62

5.4 Sunrise assertion language. 72

5.5 Sunrise assertion language types of phrases. 72

5.6 Sunrise assertion language constructor functions. 73

5.7 Sunrise programming language semantic relations. 78

5.8 Numeric Expression Structural Operational Semantics. 79

5.9 Numeric Expression List Structural Operational Semantics. . . 79

5.10 Boolean Expression Structural Operational Semantics. 80

5.11 Command Structural Operational Semantics. 81

5.12 Declaration Structural Operational Semantics. 82

5.13 Program Structural Operational Semantics. 82

5.14 Sunrise assertion language semantic functions. 83

5.15 Assertion Numeric Expression Denotational Semantics. 83
5.16 Assertion Numeric Expression List Denotational Semantics. . . 84
5.17 Assertion Boolean Expression Denotational Semantics. 84
5.18 Sunrise programming language entrance semantic relations. . . . 86
5.19 Command Entrance Semantic Relation. 87
5.20 Path Entrance Semantic Relation. 88
5.21 Sunrise programming language termination semantic relations. . 89
5.22 Command Termination Semantic Relation *C_calls_terminate*. . 89
5.23 Procedure Path Termination Semantic Relation *M_calls_terminate*. 89

6.1 Odd/Even Example Program. 98
6.2 General Rules for Total Correctness of Expressions. 100
6.3 Total Correctness of Numeric Expressions. 102
6.4 Total Correctness of Expression Lists. 103
6.5 Total Correctness of Boolean Expressions. 105
6.6 Hoare Logic for Partial Correctness. 107
6.7 General rules for Partial Correctness. 108
6.8 Entrance Logic. 112
6.9 Path Entrance Logic. 118
6.10 Additional Path Entrance Rules. 119
6.11 Call Progress Function. 120
6.12 Call Path Progress Function. 121

6.13 Command Conditional Termination Logic. 128

6.14 General rules for Command Termination. 130

6.15 Hoare Logic for Command Termination. 131

6.16 General rules for Total Correctness. 132

6.17 Hoare Logic for Total Correctness. 133

7.1 Theorems of verification of commands using the $vcg1$ function. . 162

7.2 Theorems of verification of commands using the $vcgc$ function. . 164

7.3 Theorems of verification of declarations using the $vcgd$ function. 166

7.4 Theorem of verification condition collection by $fan_out_graph_vcs$.168

7.5 Theorem of verification condition collection by $graph_vcs$. . . . 169

7.6 Theorem of verification of single recursion by $call_path_progress$. 170

7.7 Theorem of verification of all single recursion. 171

7.8 Theorem of verification of all recursion, single and multiple. . . 171

7.9 Theorem of verification of recursion by $graph_vcs$. 172

7.10 Theorem of verification of recursion by $vcgg$. 172

7.11 Theorem of verification of $vcgg$. 172

7.12 Theorem of verification of verification condition generator. . . . 173

9.1 Sunrise Theory Sizes. 228

10.1 Assertion Numeric Expression Simultaneous Substitution. . . . 236

10.2 Assertion Numeric Expression List Simultaneous Substitution. . 236

10.3 Assertion Boolean Expression Simultaneous Substitution. 237
10.4 Assertion Language Substitution Lemmas. 238
10.5 Assertion Numeric Expression Variable-for-Variable Substitution. 239
10.6 Assertion Numeric Expression List Variable-for-Variable Substitution. 239
10.7 Assertion Boolean Expression Variable-for-Variable Substitution. 240
10.8 Variables-for-Variables Substitution Creation operator $//_v$. . . . 241
10.9 Assertion Language Var-for-Var Substitution Lemmas. 242
10.10 Program Variable List Substitution. 243
10.11 Program Numeric Expression Substitution. 243
10.12 Program Numeric Expression List Substitution. 244
10.13 Program Boolean Expression Substitution. 244
10.14 Program Command Substitution. 244
10.15 Program Progress Environment Substitution. 245
10.16 Programming Language Substitution Lemmas. 245
10.17 Programming Language Substitution Equality Theorems. 246
10.18 Expression Precondition Functions. 250
10.19 Procedure Environment Well-Formedness Predicates. 255
10.20 Definition of Well-Formedness for Strings. 256
10.21 Definition of Well-Formedness for Variables. 256
10.22 Definition of Well-Formedness for Lists of Variables. 256

10.23 Definition of Not-Well-Formedness for Lists of Variables. 257

10.24 Definition of Well-Formedness for Numeric Expressions. 257

10.25 Definition of Well-Formedness for Lists of Numeric Expressions. 257

10.26 Definition of Well-Formedness for Boolean Expressions. 258

10.27 Definition of Well-Formedness for Commands. 258

10.28 Definition of Well-Formedness for Procedure Specification Syntax.259

10.29 Definition of Well-Formedness for Procedure Specification. . . . 260

10.30 Definition of Well-Formedness for Procedure Environment. . . . 261

10.31 Definition of Well-Formedness for Progress Environment. 261

10.32 Definition of Well-Formedness for Declarations. 261

10.33 Definition of Empty Progress Environment. 262

10.34 Definition of Empty Progress Environment. 262

10.35 Definition of Well-Formedness for Programs. 263

10.36 Repeated VCG verification theorems. 264

10.37 Staged command semantic relation description. 265

10.38 Staged Command Structural Operational Semantics. 266

10.39 Staged command Partial Correctness Specification. 267

10.40 Staged Well-Formed Environment Predicate for Partial Correctness. 267

10.41 Staged Command Substitution Lemmas. 267

10.42 Unstaged-to-Staged Correspondances. 268

11.1 Termination Semantic Relation *terminates*. 274

11.2 Termination Semantic Relation *Depth_calls*. 274

11.3 Theorem of existence of shallower calls. 277

11.4 Theorem of termination of shallower calls. 278

11.5 Theorem of existence of all deeper calls. 278

11.6 Sequence Generator Function *mk_sequence*. 279

11.7 Definitional property satisfied by *mk_sequence*. 279

11.8 Chain of calls induced by *mk_sequence*. 280

11.9 Infinite Recursive Descent Sequence Predicate *sequence*. 280

11.10 Recursion Expression Value Function *induct_start_num*. 281

11.11 Existence of Infinite Recursive Descent Sequence. 282

11.12 Sequence calls related by *M_calls*. 282

11.13 Sequence Precondition Maintenance. 282

11.14 Sequence Decreasing Values. 283

11.15 Sequence Occurrence Implies Limit on Occurrences. 284

11.16 Each Procedure Has Limit on Occurrences. 284

11.17 Each Procedure in Sequence is in *all_ps*. 285

11.18 Limit on All Occurrences in *all_ps*. 285

11.19 Sequence Contradiction. 286

11.20 Procedure Termination. 286

11.21 Total Correctness of Procedure Environment. 287

Acknowledgments

My earnest thanks go to Professors Rajive Bagrodia, Donald A. Martin, D. Stott Parker, and my advisor, David F. Martin, for serving on my committee.

I am grateful to The Aerospace Corporation for supporting my education with a fellowship for several years through most of my graduate studies.

Ching-Tsun Chou read early drafts of this manuscript and provided many helpful comments and suggestions. We have faced the heat of the battle together, and I have enormous respect for his intellect. I am grateful to Raymond Toal for his cheerful inspiration and solid example.

I thank my wonderful father and stepmother, Skip and Della Homeier, for too much to say, but especially for their encouragement to return to school and pursue the Ph.D. Their words sparked the desire of my heart to become a manifest reality, and have profoundly sharpened my life as a sword on the anvil.

Pastor Jim Nelson at The Church on the Way has given me consistent and faithful support over the years. From the beginning of this quest, with insight and devotion he has tenaciously watched over me, and I shall always be grateful for his protective care. I also want to thank Pastor Chip Graves, for praying and encouraging me when I needed to choose between my job and my degree.

Pastor Jack Hayford has been a staff and a covering for me. His wisdom and simple sincerity of heart have been a living model, blending biblical brilliance with tender faith and honest humility. He has an uncanny ability to speak to the heart of a situation, inevitably displaying the perspective of heaven. He has

focused me on the essentials, on integrity of heart, passion for fullness, and fierceness of commitment, as an elite trained soldier. (Isaiah 28:5-6, Zechariah 6:11, 1 Peter 5:4) The little boy who was healed from falling down is now teaching us all to go "leaping upon the mountains, skipping upon the hills." My respect and love for him know no bounds.

Verra Morgan is a shining angel, clucking like a mother hen. Her fierce, eager, combative spirit is always ready like a growling mother bear to defend her cubs, and then to cuff them when they misbehave. She has saved my neck administratively more times than I can count. I honestly do not know if I would have completed my degree if in the accident she had been taken from us. My heart overflows with thanks she was not. Verra has always lived for the success of the graduate students in her care. May she someday find a joy even greater.

Susan has been a gift beyond expectations, knitting raveled edges. She has shown me a mother's love, faith, and encouragement in the secret depths, with wisdom that few have ever seen. "For the Spirit searches all things, yes, the deep things of God. For what man knows the things of a man except the spirit of the man which is in him?" (1 Corinthians 2:10–11) She has eyes for the light in the heart of the dark, and a rod of healing held over troubled waters. My deep thanks go to her for our many rich words and woven times.

My special heartfelt appreciation goes to my advisor, Professor David F. Martin, for his supervision of this research. He has been a guiding light, an inspiration, a constant source of encouragement and approval, and the warmest of friends. He has truly been "the advisor from Heaven," in every way going beyond all that I ever conceived or could have asked for. I want to always give the same consci-

entious and diligent care he has given me. Through this long walk together, he has become a father to me, cherished beyond words. May his brilliance and his heart be recognized and rewarded.

My highest gratitude goes to someone not seen on Earth, who spun the galaxies with His fingertips, and blew the stars aflame. It was He who gave me the strength to leap each wall. Indeed every good and perfect gift comes down from above, from the Father of Lights. When opposition arose demanding I stop, He provided the way to continue and grow. When I was at the end of myself, with all my hopes at risk, He reached down and kept me from sliding into the pit, and placed my feet on a rock. He who said, "Let there be light," has shone in my heart. (2 Corinthians 4:6) All of this dissertation is only here by His hand extended in grace and peace.

Soli Deo Gloria.

Vita

1956	Born, Santa Monica, California, USA
1979	B.S. in Mathematics/Computer Science
	Summa Cum Laude
	Mathematics/Computer Science Senior Prize
	University of California, Los Angeles (UCLA)
	Los Angeles, California, USA
1979	Regents Fellowship
	University of California, Los Angeles (UCLA)
	Los Angeles, California, USA
1981	M.S. in Computer Science
	University of California, Los Angeles (UCLA)
	Los Angeles, California, USA
1981–1983	Member of the Technical Staff
	Hughes Aircraft Corporation
	El Segundo, California, USA
1983–1995	Member of the Technical Staff
	The Aerospace Corporation
	El Segundo, California, USA

1988–1994 Aerospace Graduate Fellowship
The Aerospace Corporation
El Segundo, California, USA

Publications

Peter V. Homeier and David F. Martin, "A Mechanically Verified Verification Condition Generator," *The Computer Journal*, Vol. 38, No. 2, 1995, pages 131–141.

Peter V. Homeier and David F. Martin, "Trustworthy Tools for Trustworthy Programs: A Verified Verification Condition Generator," in *Proceedings of the 7th International Workshop on Higher Order Logic Theorem Proving and its Applications*, eds. Thomas F. Melham and Juanito Camilleri, Valletta, Malta, September 19–22, 1994, Lecture Notes in Computer Science Vol. 859, Springer–Verlag, pages 269–284.

Peter V. Homeier, Thach C. Le, Y. Peter Li, and Peter C. Eggan, "DEF–CLIPS: Extensions to the CLIPS Production System Environment," in *Proceedings of Artificial Intelligence/Expert Systems Symposium*, El Segundo, CA, September 1–2, 1993.

Peter V. Homeier and Thach C. Le, "ECLIPS: An Extended CLIPS for Backward

Chaining and Goal-Directed Reasoning," in *Proceedings of the Second CLIPS Users Group Conference*, September 23–25, 1991, Houston, Texas, NASA Conference Publication 10085, Volume 2, pages 213–225.

Thach C. Le and Peter V. Homeier, "PORTABLE INFERENCE ENGINE: An Extended CLIPS for Real-Time Production Systems," in *Proceedings of the Second Annual Workshop on Space Operations Automation and Robotics*, (SOAR '88), July 20–23, 1988, NASA Conference Publication 3019, pages 187–192.

Abstract of the Dissertation

Trustworthy Tools for Trustworthy Programs: A Mechanically Verified Verification Condition Generator for the Total Correctness of Procedures

by

Peter Vincent Homeier

Doctor of Philosophy in Computer Science

University of California, Los Angeles, 1995

Professor David F. Martin, Chair

As an alternative to testing, formal proofs of a program's correctness may be constructed. The application of these techniques has been limited by the difficulty of constructing the required proofs by hand. The task of proving a program correct can be simplified and eased by a tool called a *Verification Condition Generator* (VCG), which automatically constructs a significant portion of the proof. The VCG processes programs written in the specified language, and produces as its result a set of lemmas called *verification conditions*, as the remainder left for the programmer to prove. The truth of these is intended to imply the correctness of the program. However, most VCGs that have been written have not themselves been verified, making that support unreliable.

We have written a VCG and verified its soundness, proving that if the verification conditions produced are true, then the original program is totally correct. This proof is conducted within and checked by the Higher Order Logic (HOL) mechanical proof checker, ensuring its complete soundness. The resulting verified VCG provides an effective means for proving programs totally correct with complete security.

The programming language studied contains two areas of special interest, expressions with side effects, and mutually recursive procedures, with global variables and variable and value parameters. As part of this work, we provide five program logics which together provide an axiomatic semantics for total correctness. Of the five program logics, three are fundamental inventions in this dissertation. These new logics are used to verify the correctness of expressions, the progress achieved between recursive calls of the same procedure, and the termination of procedures. All of these logics are mechanically proven within the HOL system to be sound with respect to the formal semantics of the language.

The most novel contribution of this dissertation is the discovery of a new method for proving the termination of programs with mutually recursive procedures, which is both more general and easier to use than prior proposals. In addition, VCG automation is naturally supported. This method analyzes the structure of the procedure call graph, generating verification conditions based on the cycles found.