UNIVERSITY of CALIFORNIA

Los Angeles

A Denotational Semantics Implementation System

A dissertation submitted in partial satisfaction of the
requirements for the degree Doctor of Philosophy
in Computer Science

by

Roberto da Silva Bigonha

1981

The dissertation of Roberto da Silva Bigonha is approved.

Kirby A. Baker

Daniel M. Berry

Milos D. Ercegovac

Yiannis Moschovakis

David F. Martin, Committee Chairman

University of California, Los Angeles

1981

To Mariza, my wife

To Alvaro, my father (1910-1980)

To Maria, my mother

To Alvaro Ivo, my father-in-law

To Esmeralda, my mother-in-law

TABLE OF CONTENTS

1 GENERAL PROBLEM..... 1

1.1 Definitions of Programming Language Semantics..... 1

1.1.1 The Need for Formal Semantic Definitions..... 1

1.1.2 Properties and Applications..... 2

1.1.3 Approaches to Formal Semantic Definitions..... 5

1.2 Compiler Generator Systems (CGS)..... 10

1.2.1 Introduction to CGS..... 10

1.2.2 Compiler Generator System Architecture..... 12

1.3 Problem to Be Solved..... 15

1.4 Outline of Dissertation..... 16

2 AN OVERVIEW OF MOSSES' SIS..... 18

2.1 Introduction to SIS..... 18

2.2 SIS as a Compiler Generator System..... 19

2.3 SIS as a Semantics Implementation System..... 20

2.4 The Languages of SIS..... 22

2.4.1 LAMB..... 22

2.4.2 GRAM..... 27

2.4.3 DSL..... 33

2.4.3.1 Introduction to DSL..... 33

2.4.3.2 Domain Definitions..... 34

2.4.3.3 Function Definitions..... 36

2.4.3.4 DSL Nodes..... 38

2.4.3.5 CASE Construct..... 39

2.5 Comments About SIS..... 41

3 DENOTATIONAL SEMANTICS IMPLEMENTATION SYSTEM..... 45

3.1 Introduction..... 45

3.2 Syntax Specification Language (SSL)..... 46

3.2.1 Introduction..... 46

3.2.2 Concrete Syntax..... 51

3.2.3 Abstract Syntax..... 54

3.2.4 Formal Definition of SSL..... 57

3.3 Semantic Definition Language (SDL)..... 59

3.3.1 Introduction..... 59

3.3.2 Identifiers, Basics Symbols and Domains..... 60

3.3.2.1 Identifiers..... 60

3.3.2.2 Basic Symbols..... 61

3.3.2.3 Domains..... 62

3.3.2.3.1 Built-in Domains..... 62

3.3.2.3.2 Constants..... 63

3.3.2.3.3 User Defined Domains..... 63

3.3.2.4 SDL Comments..... 67

3.3.3 SDL Expressions..... 67

3.3.3.1 Basic Expressions..... 67

3.3.3.1.1 Non-Negative Integers..... 67

3.3.3.1.2 Truth-values..... 68

3.3.3.1.3 Quotations..... 68

3.3.3.1.4 Tuples..... 69

3.3.3.1.5 Parse Tree Nodes..... 70

3.3.3.2 Inquiry Expression..... 71

3.3.3.3 Abstractions..... 73
3.3.3.4 Def-list Expressions..... 76
3.3.3.5 Conditional Expressions..... 79
3.3.3.6 CASE Expression..... 79
3.3.3.6 Updating Functions..... 80
3.3.3.8 Sequential Expressions..... 81
3.3.3.9 ACTIVATE Expression..... 83
3.3.4 Ellipsis Notation..... 83
3.3.4.1 The Approach..... 83
3.3.4.2 SDL Tuples..... 85
3.3.4.3 More Operations on Tuples 92
3.3.4.4 Updating Functions..... 97
3.3.4.5 Continuations..... 99
3.3.4.6 Conditional Forms..... 102
3.3.4.7 Examples Using Ellipses..... 103
3.3.5 Modularity Facilities..... 106
3.3.5.1 The Modularity Issue..... 106
3.3.5.2 SDL Modules..... 117
3.3.5.3 Module Structure..... 118
3.3.5.3.1 Main Module..... 118
3.3.5.3.2 External Modules..... 120
3.3.5.3.3 Internal Modules..... 124
3.3.5.4 Separate Compilation..... 127
3.3.5.5 Compilation of SDL Modules..... 132
3.3.5.6 Linking External Modules..... 141

4 SDL TYPE SYSTEM..... 143

4.1 Introduction..... 143
4.2 Type Discipline..... 144
4.2.1 The Role of Domain Variables..... 144
4.2.2 First Illustrative Example..... 147
4.2.3 Second Illustrative Example..... 152
4.2.4 Shortcomings of the Type Discipline..... 155
4.3 Outline of the Semantic Definition Style..... 156
4.4 The SDL Type Checker..... 158
4.4.1 Type of SDL..... 159
4.4.2 SDL Identifiers..... 160
4.4.3 Answers..... 163
4.4.4 Type Environment..... 166
4.4.5 Type of Modules..... 187
4.4.6 Typing SDL Sections..... 192
4.4.7 Equivalence of SDL Types..... 200
4.4.8 SDL Type Polymorphism..... 210
4.4.8.1 Explication..... 211
4.4.8.2 Instantiation..... 212
4.4.8.3 Unification..... 213
4.4.8.4 Recursive Domain Variables..... 216
4.4.8.5 Particularization of Domains..... 219
4.4.8.6 Type of Functional Applications..... 220
4.4.8.7 Unique Name Generation..... 221
4.4.9 Typing of Patterns..... 233
4.4.10 Type of Expressions..... 239
4.4.11 Typing of Definitions..... 254
4.4.12 Type Checker and Ellipsis..... 266

5 THE SEMANTICS OF SDL 279
5.1 Outline of Method 279
5.2 Module Structure 281
5.2.1 The Main Module 281
5.2.2 The Environment 283
5.2.3 Semantics of Modules 287
5.2.4 Semantics of Sections 295
5.2.5 Link Editor 300
5.2.6 Semantics of Expressions 308
5.2.7 Semantics of Patterns 329
5.2.8 Semantics of Definitions 334
5.3 SDL Definitions and the System 341

6 A METHODOLOGY FOR STRUCTURING DENOTATIONAL SEMANTICS 344
6.1 Introduction 344
6.2 An Example Language 351
6.3 Concrete and Abstract Syntax of ASPLE 352
6.4 ASPLE Type Checker 354
6.4.1 Outline of Method 354
6.4.2 Machine Context 357
6.4.3 Type Environment 357
6.4.4 Module Organization 358
6.4.5 SDL Definition of ASPLE Type Checker 358
6.5 The Semantics of ASPLE 368
6.5.1 Outline of Method 368
6.5.2 The Run-time Environment 369

6.5.3 The Abstract Machine..... 370

6.5.4 The Concrete Machine..... 373

6.5.5 The SDL Semantics of ASPLE..... 373

6.6 Conclusion..... 382

7 CONCLUSIONS AND FUTURE WORK..... 384

A. SSL SYNTAX OF SSL..... 387

B. SDL SEMANTIC DEFINITION OF SSL..... 391

C. SSL SYNTAX OF SDL..... 413

REFERENCES..... 420

TABLE OF FIGURES

Figure 2.1 : Compiler-Compiler Structure..... 21

Figure 2.2 : Use of (DSL) LET and ALSO Forms..... 38

Figure 2.3 : General Form of (DSL) CASE Construct..... 40

Figure 2.4 : Code for (DSL) CASE Construct..... 40

Figure 3.1 : SDL Keywords..... 61

Figure 3.2 : Special Symbols..... 62

Figure 3.3 : General Form of CASE..... 79

Figure 3.4 : CASE With Ellipsis..... 90

Figure 3.5 : Definition of #elem..... 92

Figure 3.6 : Definition of #set..... 95

Figure 3.7 : Definition of #for 97

Figure 3.8 : Definition of #update..... 99

Figure 3.9 : Definition of #nest1..... 101

Figure 3.10: Definition of #nest#2..... 101

Figure 3.11: Definition of #nest'..... 102

Figure 3.12: Definition of #cond..... 103

Figure 3.13: Skeleton of an External Module..... 122

Figure 3.14: Nesting of Modules..... 125

Figure 3.15: Circularly Dependent Modules..... 128

Figure 3.16: Bodies of Circularly Dependent Modules..... 129

Figure 3.17: Code for Module Bodies..... 130

Figure 3.18: A Main Module Skeleton..... 132

Figure 3.19: Non-circularly Dependent Modules..... 136

Figure 3.20: An External Module Body..... 137

Figure 3.21: A Linked Code Skeleton..... 138

ACKNOWLEDGEMENTS

I wish to express my gratitude to my advisor, Professor David F. Martin, for his valuable guidance, encouragement, continual moral support and meticulous reading through countless drafts of this work.

Appreciation goes to Professors Kirby A. Baker, Daniel M. Berry, Milos D. Ercegovac, and Yiannis Moschovakis for serving in the doctoral committee.

I am also grateful to Universidade Federal de Minas Gerais and Coordenação do Aperfeiçoamento de Pessoal de Nível Superior (CAPES) for sponsoring this research.

Special thanks go to Mariza, my wife, who sacrificed her own professional career so that I could come to UCLA. Without her love, understanding and constant moral support this research would not have been possible.

At last but not least, I would like to thank Professor José Antonio de Faria and Alvaro Theotonio Andrade da Silva for their friendship and for taking care of my Brazilian affairs while I was absent from Brazil.

VITA

May 19, 1948 — Born, Ubá, Minas Gerais, Brazil.

1971 — B.S., Escola de Engenharia, Universidade Federal de Minas Gerais, Belo Horizonte, Minas Gerais, Brazil.

1972 - 1973 — Teaching Assistant, Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, Brazil.

1975 — M.S., Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, Brazil.

1973 - 1976 — Lecturer, Departamento de Ciência da Computação, Universidade Federal de Minas Gerais, Belo Horizonte, Minas Gerais, Brazil.

1977 - — Assistant Professor, Departamento de Ciência da Computação, Universidade Federal de Minas Gerais, Belo Horizonte, Minas Gerais, Brazil.

ABSTRACT OF THE DISSERTATION

A Denotational Semantics Implementation System

by

Roberto da Silva Bigonha

Doctor of Philosophy in Computer Science

University of California, Los Angeles, 1981

Professor David F. Martin, Chairman

A system for formulating, testing and implementing denotational semantic definitions is designed and formally defined. Mosses' Semantics Implementation System (SIS) has been taken as a starting-point. SIS has been modified and extended in order to increase its expressive power and to enhance reliability and readability of denotational definitions.

The main work is toward the definition and specification of a machine-processable Denotational Semantic Language (SDL) for conveying modular denotational semantic definitions. In addition to the usual notation of semantic equations, SDL contains specially designed constructs to facilitate modularization and data abstraction. Moreover, SDL embodies some convenient notations such as the ellipsis (...) and "updating" functions.

A polymorphic type discipline has been adopted to give SDL the advantages of strongly typed languages such as Algol 68 while benefiting from the flexibility of defining functions that accept parameters of

various types.

Both the formal definition of the SDL type system and of the semantics of SDL are written in SDL itself.

In addition to SDL, our system incorporates two other languages: SSL for syntax specifications and LAMB, the semantic base language borrowed from Mosses' SIS.

Finally, a data-abstraction-based methodology for formulating structured denotational definitions is proposed and illustrated.