

AMAST Series in Computing Vol. 1

ALGEBRAIC SPECIFICATION TECHNIQUES AND TOOLS FOR SOFTWARE DEVELOPMENT

The ACT Approach

Ingo Claßen

Hartmut Ehrig

Dietmar Wolz

*Institute for Software and Theoretical Computer Science
Technical University of Berlin, Germany*



World Scientific

Singapore • New Jersey • London • Hong Kong

Contents

Foreword	v
Preface	vii
Introduction	xiii
1 Algebraic Specification as Formal Method for Software Development	1
1.1 General Aspects of Software Development	1
1.1.1 Overview of Requirements for Software Systems	2
1.1.2 Conceptual Software Development Process	3
1.1.3 Role of Stages and Steps and Support by Formal Methods . .	5
1.1.4 Software Correctness and Verification	7
1.2 Basic Algebraic Specification Concepts of the ACT Approach	8
1.2.1 Algebraic Specifications	8
1.2.2 Parameterized Specifications	10
1.2.3 Module Specifications	11
1.2.4 Transformation of Specifications	13
1.3 Software Development Methodology of the ACT Approach	14
1.3.1 Conceptual Development Stages in the ACT Approach	15
1.3.2 Horizontal Structuring, Internal Correctness and Compositionality	16
1.3.3 Vertical Development and Correctness	18
1.3.4 Tool Support for the ACT Approach	20
1.3.5 Summary of the ACT Approach	22
2 ACT ONE — A Language for Parameterized Specifications	23
2.1 Basic Language Units	25
2.1.1 Unparameterized Types	25
2.1.2 Parameters	30
2.1.3 Parameterized Types	33
2.2 Structuring Mechanisms	40
2.2.1 Extension and Union	41

2.2.2	Renaming and Actualization	42
2.2.3	ACT Text	46
2.3	Language Description	47
2.3.1	Syntax	47
2.3.2	Structure Resolution	51
2.3.3	Syntactical Context Conditions	55
2.3.4	Semantics	57
2.3.5	Semantical Context Conditions	59
2.3.6	Correctness and Compositionality	60
2.4	Case Study: Syntax Directed Editor	61
2.4.1	Overall Requirement Definition	62
2.4.1.1	An Example	63
2.4.1.2	The User Machine	65
2.4.1.3	User Interface	66
2.4.1.4	Programming Language	67
2.4.2	Functional Requirement Specification	68
2.4.2.1	Model Data	69
2.4.2.2	Model Operations	70
2.4.3	Design Specification	72
2.4.3.1	Basic Data Types	73
2.4.3.2	Application Data Types	83
2.4.3.3	Input/Output Data Types	91
2.4.3.4	User Interface Data Types	93
2.4.4	Results and Limitations of the Case Study	97
2.4.4.1	Model	98
2.4.4.2	Specification	98
2.4.4.3	Prototyping	99
2.4.4.4	Correctness and Compositionality	99
2.4.4.5	Limitations	100
	Algebraic Specification of Modular Systems	101
3.1	Modularity of Software Systems	102
3.1.1	Principles of Modularization	102
3.1.2	Concepts of Modularity	103
3.1.3	Integration, Customization, Reusability, and Extensibility of Software Components	105
3.1.4	Algebraic Concepts for Modularity	107
3.1.5	Conceptual Links	108
3.2	Algebraic Module and Language Concepts	110
3.2.1	Module Specifications	110
3.2.2	Module Interconnection	111
3.2.3	Module Language Concepts	112

3.3	Modular Design of a Material Requirements Planning System	114
3.3.1	Material Requirements Planning (MRP)	115
3.3.2	MRP Module Specification	115
3.3.3	MRP Module Interconnection	125
3.3.4	Summary of the Modular Design of MRP	126
4	The ACT Environment	129
4.1	Syntax and Static Semantics Checker	131
4.2	User Interface for Prototyping	133
4.2.1	Evaluation of Terms and Goals	133
4.2.2	Information about the specification	134
4.2.3	Visualization of Terms	134
4.2.4	Prototype Execution of the Syntax Directed Editor	138
4.3	Evaluation of Terms and Goals by LANAM	145
4.3.1	Narrowing and Rewriting	147
4.3.2	Criteria for Evaluable Specifications	150
4.3.2.1	Sufficient Completeness and Consistency	150
4.3.2.2	Criteria for the applicability of narrowing and rewriting	151
4.3.3	Evaluation Strategies	152
4.3.3.1	Advantages of the lazy evaluation strategy	153
4.3.3.2	Examples for lazy narrowing	154
4.3.4	The lazy narrowing algorithm	157
4.3.4.1	Terminology	158
4.3.4.2	Operational Behavior	158
4.3.5	Implementation of Lazy Narrowing by Compilation of Rules .	159
4.3.6	Compilation vs. Interpretation of Data Type Specifications .	161
4.3.6.1	Compilation vs. Interpretation of Abstract Machine Code	161
4.3.6.2	Alternative Approaches to Implement Lazy Narrowing	162
4.3.6.3	Performance	162
4.3.7	Conclusion	162
4.4	Termination, Completeness and Consistency Checker	163
4.4.1	Criteria Checked by the Tool	164
4.4.2	Violations of the Checked Criteria	166
4.4.3	Example for data type checking	167
4.4.4	Implementation, Limitations and Performance	172
4.4.5	Persistency	173
4.4.6	Completion	174
4.4.7	Conclusion	174

A	Formal Notions of Algebraic Specifications	177
A.1	Algebraic Specifications	178
A.1.1	Signatures and Specifications	178
A.1.2	Algebras and Homomorphisms	179
A.1.3	Term Algebras, Initial, and Free Algebras	180
A.1.4	Congruence of Terms and Quotient Term Algebra	181
A.1.5	Semantics and Correctness	181
A.1.6	Constraints and Logic of Constraints	181
A.1.7	Specifications with Constraints	182
A.2	Parameterized Specifications	183
A.2.1	Syntax, Semantics and Correctness of Parameterized Specifications	183
A.2.2	Actualization, Amalgamation, and Extension	184
A.3	Module Specifications	186
A.3.1	Syntax, Semantics and Correctness of Module Specifications	186
A.3.2	Operations on Module Specifications	188
A.4	Operational Semantics for Conditional Equations	193
A.4.1	The Conditional Equational Calculus	193
A.4.2	Narrowing	194
B	User Manuals for Tool Support	197
B.1	Syntax and Static Semantics Checker	197
B.2	User Interface for Prototyping	199
B.2.1	Evaluation of Terms and Goals	199
B.2.2	Parsing	202
B.2.3	Queries	204
B.2.4	Internal Generation of Equations	205
B.2.5	Completeness, Consistency and Termination Checking	206
B.2.6	Parameter Setting	207
B.2.7	Output Transformation	208
B.2.8	Command Syntax	209
B.3	<i>GVT</i> — Graphical Visualization of Terms	211
B.3.1	Syntax of <i>GVT</i>	211
B.3.2	Examples	215
B.3.3	<i>GVT</i> System Usage	217
	Bibliography	221
	Subject Index	231