

# Contents

<b>Preface</b> .....	vii
<b>0 Introduction</b> .....	1
0.1 Modelling software systems as algebras .....	1
0.2 Specifications .....	5
0.3 Software development .....	8
0.4 Generality and abstraction .....	10
0.5 Formality .....	13
0.6 Outlook .....	14
<b>1 Universal algebra</b> .....	15
1.1 Many-sorted sets .....	15
1.2 Signatures and algebras .....	19
1.3 Homomorphisms and congruences .....	22
1.4 Term algebras .....	27
1.5 Changing signatures .....	32
1.5.1 Signature morphisms .....	33
1.5.2 Derived signature morphisms .....	36
1.6 Bibliographical remarks .....	38
<b>2 Simple equational specifications</b> .....	41
2.1 Equations .....	42
2.2 Flat specifications .....	44
2.3 Theories .....	49
2.4 Equational calculus .....	53
2.5 Initial models .....	57
2.6 Term rewriting .....	65
2.7 Fiddling with the definitions .....	71
2.7.1 Conditional equations .....	72
2.7.2 Reachable semantics .....	74
2.7.3 Dealing with partial functions: error algebras .....	78

2.7.4	Dealing with partial functions: partial algebras . . . . .	83
2.7.5	Partial functions: order-sorted algebras . . . . .	86
2.7.6	Other options . . . . .	90
2.8	Bibliographical remarks . . . . .	93
<b>3</b>	<b>Category theory . . . . .</b>	<b>97</b>
3.1	Introducing categories . . . . .	99
3.1.1	Categories . . . . .	99
3.1.2	Constructing categories . . . . .	105
3.1.3	Category-theoretic definitions . . . . .	109
3.2	Limits and colimits . . . . .	111
3.2.1	Initial and terminal objects . . . . .	112
3.2.2	Products and coproducts . . . . .	113
3.2.3	Equalisers and coequalisers . . . . .	115
3.2.4	Pullbacks and pushouts . . . . .	116
3.2.5	The general situation . . . . .	119
3.3	Factorisation systems . . . . .	123
3.4	Functors and natural transformations . . . . .	127
3.4.1	Functors . . . . .	128
3.4.2	Natural transformations . . . . .	135
3.4.3	Constructing categories, revisited . . . . .	139
3.5	Adjoint functors . . . . .	144
3.5.1	Free objects . . . . .	144
3.5.2	Left adjoints . . . . .	145
3.5.3	Adjunctions . . . . .	150
3.6	Bibliographical remarks . . . . .	152
<b>4</b>	<b>Working within an arbitrary logical system . . . . .</b>	<b>155</b>
4.1	Institutions . . . . .	158
4.1.1	Examples of institutions . . . . .	161
4.1.2	Constructing institutions . . . . .	180
4.2	Flat specifications in an arbitrary institution . . . . .	187
4.3	Constraints . . . . .	194
4.4	Exact institutions . . . . .	198
4.4.1	Abstract model theory . . . . .	205
4.4.2	Free variables and quantification . . . . .	209
4.5	Institutions with reachability structure . . . . .	213
4.5.1	The method of diagrams . . . . .	216
4.5.2	Abstract algebraic institutions . . . . .	218
4.5.3	Liberal abstract algebraic institutions . . . . .	219
4.5.4	Characterising abstract algebraic institutions that admit reachable initial models . . . . .	221
4.6	Bibliographical remarks . . . . .	223

<b>5</b>	<b>Structured specifications</b>	229
5.1	Specification-building operations	230
5.2	Towards specification languages	237
5.3	An example	241
5.4	A property-oriented semantics of specifications	245
5.5	The category of specifications	249
5.6	Algebraic laws for structured specifications	253
5.7	Bibliographical remarks	257
<b>6</b>	<b>Parameterisation</b>	259
6.1	Modelling generic modules	260
6.2	Specifying generic modules	270
6.3	Parameterised specifications	276
6.4	Higher-order parameterisation	280
6.5	An example	287
6.6	Bibliographical remarks	290
<b>7</b>	<b>Formal program development</b>	293
7.1	Simple implementations	294
7.2	Constructor implementations	302
7.3	Modular decomposition	309
7.4	Example	316
7.5	Bibliographical remarks	322
<b>8</b>	<b>Behavioural specifications</b>	325
8.1	Motivating example	326
8.2	Behavioural equivalence and abstraction	329
8.2.1	Behavioural equivalence	330
8.2.2	Behavioural abstraction	335
8.2.3	Weak behavioural equivalence	337
8.3	Behavioural satisfaction	340
8.3.1	Behavioural satisfaction vs. behavioural abstraction	343
8.4	Behavioural implementations	348
8.4.1	Implementing specifications up to behavioural equivalence	348
8.4.2	Stepwise development and stability	350
8.4.3	Stable and behaviourally trivial constructors	352
8.4.4	Global stability and behavioural correctness	357
8.4.5	Summary	365
8.5	To partial algebras and beyond	366
8.5.1	Behavioural specifications in <b>FPL</b>	366
8.5.2	A larger example	373
8.5.3	Behavioural specifications in an arbitrary institution	384
8.6	Bibliographical remarks	396

<b>9</b>	<b>Proofs for specifications</b>	401
9.1	Entailment systems	402
9.2	Proof in structured specifications	415
9.3	Entailment between specifications	429
9.4	Correctness of constructor implementations	438
9.5	Proof and parameterisation	443
9.6	Proving behavioural properties	453
9.6.1	Behavioural consequence	454
9.6.2	Behavioural consequence for specifications	465
9.6.3	Behavioural consequence between specifications	469
9.6.4	Correctness of behavioural implementations	473
9.6.5	A larger example, revisited	475
9.7	Bibliographical remarks	482
<b>10</b>	<b>Working with multiple logical systems</b>	485
10.1	Moving specifications between institutions	486
10.1.1	Institution semi-morphisms	487
10.1.2	Duplex institutions	491
10.1.3	Migrating specifications	493
10.2	Institution morphisms	501
10.3	The category of institutions	511
10.4	Institution comorphisms	518
10.5	Bibliographical remarks	530
	<b>Bibliography</b>	533
	<b>Index of categories and functors</b>	553
	<b>Index of institutions</b>	555
	<b>Index of notation</b>	557
	<b>Index of concepts</b>	563

Foundations of Algebraic Specification and Formal  
Software Development

Sannella, D.; Tarlecki, A.

2012, XVI, 584 p., Hardcover

ISBN: 978-3-642-17335-6