

Contents

List of contributors	xv
Models for concurrency	1
<i>Glynn Winskel and Mogens Nielsen</i>	
1 Introduction	1
2 Transition systems	7
2.1 A category of transition systems	7
2.2 Constructions on transition systems	10
3 A process language	20
3.1 Operational semantics (version 1)	22
3.2 Operational semantics (version 2)	23
3.3 An example	26
4 Synchronisation trees	28
5 Languages	32
6 Relating semantics	34
7 Trace languages	36
7.1 A category of trace languages	37
7.2 Constructions on trace languages	40
8 Event structures	41
8.1 A category of event structures	44
8.2 Domains of configurations	45
8.3 Event structures and trace languages	47
9 Petri nets	61
9.1 A category of Petri nets	65
9.2 Constructions on nets	66
10 Asynchronous transition systems	70
10.1 Asynchronous transition systems and trace languages	73
10.2 Asynchronous transition systems and nets	75
10.3 Properties of conditions	91
11 Semantics	102
11.1 Embeddings	102
11.2 Labelled structures	106
11.3 Operational semantics	109
12 Relating models	118
13 Notes	122
Acknowledgements	128
References	129

Appendix A: A basic category	135
Appendix B: Fibred categories	135
Appendix C: Operational semantics—proofs	139

Concrete process algebra 149

J. C. M. Baeten and C. Verhoef

1	Introduction	
2	Concrete sequential processes	150
	2.1 Introduction	152
	2.2 Basic process algebra	152
	2.3 Recursion in BPA	152
	2.4 Projection in BPA	167
	2.5 Deadlock	169
	2.6 Empty process	181
	2.7 Renaming in BPA	184
	2.8 The state operator	190
	2.9 The extended state operator	197
	2.10 The priority operator	201
	2.11 Basic process algebra with iteration	204
	2.12 Basic process algebra with discrete relative time	217
	2.13 Basic process algebra with other features	220
	2.14 Decidability and expressiveness results in BPA	224
3	Concrete concurrent processes	226
	3.1 Introduction	228
	3.2 Syntax and semantics of parallel processes	229
	3.3 Extensions of PA	229
	3.4 Extensions of PA_{δ}	235
	3.5 Syntax and semantics of communicating processes	241
	3.6 Extensions of ACP	244
	3.7 Decidability and expressiveness results in ACP	251
4	Further Reading	255
	References	260
		260

Correspondence between operational and denotational semantics: the full abstraction problem for PCF

C.-H. L. Ong

1	Introduction	
	1.1 Relating operational and denotational semantics	270
	1.2 Full abstraction problem for PCF	270
	1.3 Quest for a solution: a survey	274
		276

1.4	Organization of the chapter	280
1.5	A selected bibliography	281
2	A programming language for computable functions	282
2.1	The programming language PCF	283
2.2	Syntax of the reduction system $\lambda_{\vec{v}}(\mathbf{A})$	284
2.3	Reduction rules of the system $\lambda_{\vec{v}}(\mathbf{A})$	287
2.4	Operational properties of $\lambda_{\vec{v}}(\mathbf{A})$	288
2.5	Böhm trees and the Syntactic Continuity Theorem	292
2.6	Sequentiality and stability	295
2.7	The programming language PCF	299
3	Operational and denotational semantics of PCF	301
3.1	Context Lemma and observational extensionality	302
3.2	Denotational models	304
3.3	Adequacy	308
3.4	Order-extensional, continuous, fully abstract model	313
3.5	Full abstraction and non-full abstraction results	319
4	Towards a characterization of PCF-sequentiality	325
4.1	Concrete data structures and sequential functions	327
4.2	dI-domains and stable functions	335
4.3	Sequential algorithms	339
4.4	Observable algorithms and PCF with error values	342
4.5	Towards a characterization of sequentiality	345
	Acknowledgements	349
	References	350

Effective algebras 357

V. Stoltenberg-Hansen and J. V. Tucker

1	Introduction	358
1.1	Computing in algebras	359
1.2	Examples of countable algebras	361
1.3	Examples of uncountable algebras	363
1.4	Definitions of computable algebras	364
1.5	General algebraic framework for computations on algebras	366
1.6	Historical notes on computable algebra	368
1.7	Objectives and structure of the chapter	371
1.8	Prerequisites	372
2	Computable algebras	372
2.1	Preliminaries on algebras	373
2.2	Computable, semicomputable, and cosemicomputable algebras	377
2.3	Invariance	387
2.4	A few computable constructions	397
2.5	Concluding remarks	402
3	Algebraic characterisations of computable algebras	402
3.1	Computable data types and their specification	403

3.2	Equational specifications	411
3.3	Adequacy theorem	421
3.4	Equational specifications and term rewriting	427
3.5	Proof of First Completeness Theorem	433
3.6	Concluding remarks	444
4	Domains and approximations for topological algebras	445
4.1	Approximation structures and topologies	446
4.2	Representing topological algebras by structured domains	451
4.3	Inverse limits and ultrametric algebras	456
4.4	Total elements in domains	467
4.5	Representability of locally compact Hausdorff algebras	472
5	Effective domains	479
5.1	Basic theory	480
5.2	Constructive subdomains	485
5.3	Algebras effectively approximable by domains	489
5.4	The Myhill–Shepherdson Theorem	501
5.5	The Kreisel–Lacombe–Shoenfield Theorem	505
	References	512

Abstract interpretation: a semantics-based tool for program analysis

527

Neil D. Jones and Flemming Nielson

1	Introduction	
1.1	Goals and motivations	528
1.2	Relation to program verification and transformation	528
1.3	The origins of abstract interpretation	535
1.4	A sampling of data-flow analyses	535
1.5	Outline	536
2	Basic concepts and problems to be solved	538
2.1	A naive analysis of the simple program	539
2.2	Accumulating semantics for imperative programs	540
2.3	Correctness and safety	542
2.4	Scott domains, lattice duality, and meet versus join	548
2.5	Abstract values viewed as relations or predicates	555
2.6	Important points from earlier sections	556
2.7	Towards generalizing the Cousot framework	560
2.8	Proving safety by logical relations	561
3	Abstract interpretation using a two-level metalanguage	565
3.1	Syntax of metalanguage	568
3.2	Specification of analyses	569
3.3	Correctness of analyses	575
3.4	Induced analyses	588
3.5	Expected forms of analyses	595
3.6	Extensions and limitations	604
		609

Contents

xiii

4	Other analyses, language properties, and language types	610
4.1	Approaches to abstract interpretation	611
4.2	Examples of instrumented semantics	614
4.3	Analysis of functional languages	616
4.4	Complex abstract values	621
4.5	Abstract interpretation of logic programs	623
5	Glossary	627
	References	629

Index

637