Appendix 0: Alphabets

Observables

	SL	LabP	LL	ACP	CSP	DF	HL	DL
stable (ok, ok')	√	√	√		√	√	√	√
waiting ($wait$, $wait'$)				√	√	√		√
control (l, l')		√						
program variables (v, v')	√	√					√	√
question (q), answers (a')			√					
trace (tr, tr')				√	√	√		
refusal (ref, ref')				√	√	√		
procedure variables (p, p')							√	√

- **SL**: Sequential programming language defined in Chapter 5.
- **LabP**: Labelled programs defined in Chapter 6.
- **LL**: Logic programming language defined in Section 7.6.
- **ACP**: Algebra of communicating processes defined in Section 8.1.
- **CSP**: Communicating Sequential Processes defined in Section 8.2.
- **DF**: Data flow processes defined in Section 8.3.
- **HL**: High order language defined in Chapter 9.
- **DL**: Declarative programming language defined in Section 9.4.

Appendix 1: Shared Variables

	Merge relation M	Atomic actions $\mathcal{A}$
stable (ok') (7.1)	$(0.ok \wedge 1.ok) \Rightarrow ok'$	
resource (r) (7.2)	$r' = 0.r + 1.r - r$	$r := r + e$
log (out) (7.2)	$(out' - out) \in$	$out := out^\frown < e >$
	$(0.out - out)\|\|\|(1.out - out)$	
clock ($clock$) (7.2)	$clock' = max(0.clock,\ 1.clock)$	$clock := clock + 1$
input (in, c) (7.2)	$(0.c = 1.c)_\perp; c := 0.c$	$m,\ c := in_c,\ c + 1$
output (out, c) (7.2)	$\bigwedge_{c \le i < 0.c}\ out_i := (0.out_i \| 1.out_i);$	$out_c,\ c := e,\ c + 1$
	$(0.c = 1.c)_\perp; c := 0.c$	
array (A) (7.4)	$A' = A \oplus (r_0 \lhd 0.A) \oplus (s_0 \lhd 1.A)$	$A[i] := e,\ x := f(A[j])$
answers (a') (7.6)	$a' = 0.a^\frown 1.a$	$a := a^\frown < c >$
	$a' \in (0.a\|\|\|1.a)$	
waiting ($wait'$)	$wait' = 0.wait \vee 1.wait$	$wait := true$
(8.1 and 8.2)		$wait := false$
trace (tr')	$(tr' - tr) \in (0.tr - tr)\|\|(1.tr - tr)$	$tr := tr^\frown < c >$
(8.1 and 8.2)	$(tr' - tr) \in (0.tr - tr)\|\|\|(1.tr - tr)$	
	$(tr' - tr) \downarrow \mathcal{A}P = (0.tr - tr) \wedge$	
	$(tr' - tr) \downarrow \mathcal{A}Q = (1.tr - tr)$	
refusal (ref')	$ref' = 0.ref \cap 1.ref$	$\mathbf{true} \vdash (ref' \cap X = \{\})$
(8.1 and 8.2)	$ref' = 0.ref \cup 1.ref$	
program (X') (9.4)	$X' \sqsupseteq (0.X^\bullet \| 1.X)$	$X' \sqsupseteq P$

Appendix 2: Primitives

	Abort	Skip	Stop	Miracle
SL **LabP** **LL**	**true**	$II_{\{v\}} = \mathbf{true} \vdash (v' = v)$		$\neg\, ok$
ACP **CSP** **DF**	**R(true)**	$\mathbf{R}(\exists ref' \bullet II_{\{tr, ref, wait\}})$	$\mathbf{R}(\mathbf{true} \vdash wait' \wedge tr' = tr)$	$\mathbf{R}(\neg\, ok)$
DL	**W(true)**	$\mathbf{W}(II_{\{v\}})$	$wait := true$	$\mathbf{W}(\neg\, ok)$
HL	**true**	$II_{\{p\}} = \mathbf{true} \vdash (p' \sqsupseteq p)$		$\neg\, ok$

where **R** and **W** are defined by

$$\mathbf{R}(X(tr, tr')) \;=_{df} (tr \leq tr') \;\wedge\; (II_{\{tr, ref, wait\}} \lhd wait \rhd (\bigvee_s X(s, s^\frown(tr' - tr))))$$
$$\mathbf{W}(X) \;=_{df}\; (STOP \lhd wait \rhd X)$$

Appendix 3: Healthiness Conditions

Sequential programming language (Section 3.2 and Example 4.1.21)

H1 $P \;=\; (\neg ok \lor P)$

H2 $[P[false/ok'] \Rightarrow P[true/ok']]$

H3 $P \;=\; P; I\!I$

H4 $P; \mathbf{true} \;=\; \mathbf{true}$ □

Reactive processes (Section 8.0)

R1 $P \;=\; P \land (tr \leq tr')$

R2 $P(tr, tr') \;=\; P(<>, tr' - tr)$

R3 $P \;=\; I\!I_{\{tr, ref, wait\}} \lhd wait \rhd P$ □

ACP (Section 8.1)

R1–R3

ACP1 $P \land (tr' = tr) \;\Rightarrow\; wait'$ □

CSP (Section 8.2)

R1–R3

CSP1 $P \;=\; \neg ok \wedge (tr \leq tr') \;\vee\; P$

CSP2 $P \;=\; P; ((ok \Rightarrow ok') \;\wedge\; (tr' = tr) \;\wedge\; \ldots \;\wedge\; (ref' = ref))$

CSP3 $P \;=\; SKIP; P$

CSP4 $P \;=\; P; SKIP$

CSP5 $P \;=\; P|||SKIP$

Data flow processes (Section 8.3)

R1–R3

CSP1–CSP5

DF1 $P \;=\; IN \gg P \gg OUT$ □

High order language (Section 9.1)

P1 $\Pi_{\{p\}}; P \;=\; P$

P2 $P; \Pi_{\{p\}} \;=\; P$ □

Definedness function (Section 9.3)

D1 $\mathcal{D}e[false/\mathcal{D}x] \Rightarrow \;(\mathcal{D}e)[true/\mathcal{D}x]$

D2 $(\mathcal{D}e)[false/\mathcal{D}x] \;\Rightarrow\; \forall x, \mathcal{D}x \bullet \mathcal{D}(e)$

D3 $\mathcal{D}(\mathcal{D}e)$ □

Declarative language (Section 9.4)

W1 $P \;=\; \Pi \lhd wait \rhd P$

W2 $P \;=\; P \;\vee\; (P \wedge wait' \wedge v' \neq v); \mathbf{true}$ □

Bibliography

[1] M. Abadi and L. Lamport. Composing specifications. *ACM Transactions on Programming Languages and Systems*, Vol 15(1): 73–132, (1993).

[2] M. Abadi and L. Lamport. Conjoining specifications. *ACM Transactions on Programming Languages and Systems*, Vol 17(3): 507–534, (1995).

[3] M.L. Abell and P. Braselton. *Maple V Handbook.* Academic Press, (1994).

[4] S. Abramsky and R. Jagadeesan. Games and full completeness for multiplicative linear logic. *Journal of Symbolic Logic*, Vol 59: 543–574, (1994).

[5] S. Abramsky and C.-H. Ong. Full abstraction in the lazy lambda calculus. *Information and Computation*, Vol 105: 159–267, (1993).

[6] S. Abramsky, D.M. Gabbay and T.S.E. Maibaum. *Semantic Structures.* Handbook of Logics in Computer Sciences, Oxford University Press, (1994).

[7] J.-R. Abrial. *The B-Book: Assigning Programs to Meanings.* Cambridge University Press, (1996).

[8] G. Agha. *Actors: A Model of Concurrent Computation in Distributed Systems.* The MIT Press, (1986).

[9] G. Agha and C. Hewitt. Actors: A conceptual foundation for object-oriented programming. In P. Wegner and B. Shrivers (eds): *Research Direction in Object-oriented Programming*, 49–74, The MIT Press, (1987).

[10] R. Alur and T.A. Henzinger. Logics and models of real time: a survey. *Lecture Notes in Computer Science*, Vol 600: 74–106, Springer-Verlag, (1992).

[11] K.R. Apt and E.-R. Olderog. *Verification of Sequential and Concurrent Programs.* Springer-Verlag, (1991).

[12] K. Arnold and J. Gosling. *The Java Programming Language.* Addison-Wesley, (1996).

[13] R.J.R. Back and K. Sere. Stepwise refinement of action systems. *Lecture Notes in Computer Science*, Vol 375: 115–138, Springer-Verlag, (1989).

[14] R.J.R. Back and J. von Wright. Refinement calculus, Part I: Sequential nondeterministic programs. *Lecture Notes in Computer Science*, Vol 430: 42–66, Springer-Verlag, (1989).

[15] R.C. Backhouse. Factor theory revisited. Technical Report of Edinhoven University of Technology, (1990).

[16] R.C. Backhouse and P. Hoogendijk. Elements of the relational theory of data types. *Lecture Notes in Computer Science*, Vol 755, Springer-Verlag, (1993).

[17] J. Backus. Can programming be liberated from the von Neumann style? A functional style and its algebra of programs. *Communications of the ACM*, Vol 21(8): 613–641, (1978).

[18] J.C.M. Baeten and W.P. Weijland. *Process Algebra*. Cambridge Tracts in Theoretical Computer Science, Cambridge University Press, (1990).

[19] J.W. de Bakker. Semantics and termination of nondeterministic recursive programs. *Proceedings of 3rd International Colloquium on Automata, Language and Programming*, 436–477, Edinburgh University Press, (1976).

[20] J.W. de Bakker. *Mathematical Theory of Program Correctness.* Prentice Hall, (1980).

[21] J.W. de Bakker and W.-P. de Roever. A calculus for recursive program schemes. *Proceedings of 1st International Colloquium on Automata, Language and Programming*, 167–196, North-Holland, (1972).

[22] M. Barr and C. Wells. *Category Theory for Computing Science,* Second Edition. Prentice Hall, (1995).

[23] J. A. Bergstra and J. W. Klop. Algebra of communicating processes with abstraction. *Theoretical Computer Science*, Vol 37(1): 77–121, (1985).

[24] G. Berry. Esterel on hardware. *Philosophical Transactions of the Royal Society of London*, Series A, Vol 339: 87–104, (1992).

[25] G. Berry and G. Gonthier. The Esterel synchronous programming language: design, semantics, implementation. Rapport de Recherche 842, INRIA, (1988).

[26] R.J. Bird and P. Wadler. *Introduction to Functional Programming.* Prentice Hall, (1988).

[27] R.S. Bird. Transformational programming and the paragraph problem. *Science of Computer Programming*, Vol 6: 159–189, (1986).

[28] R.S. Bird and L. Meertens. Two exercises found in a book on algorithmics. In L.G.L.T. Meertens (ed): *Program Specification and Transformations*, 451–457, Elsevier Science Publishers B.V., North-Holland, (1987).

[29] D. Bjørner. A ProCoS project description. *Proceedings of International Conference on AI and Robotics*, North-Holland, (1989).

[30] R.S. Boyer and J.S. Moore. *A Computational Logic Handbook.* Academic Press, (1988).

[31] J.D. Brock and W.B. Ackermann. Scenarios: a model of nondeterministic computation. *Lecture Notes in Computer Science*, Vol 107: 252–267, Springer-Verlag, (1982).

[32] S.D. Brookes, C.A.R. Hoare and A.W. Roscoe. A theory of communicating sequential processes. *Journal of the ACM*, Vol 31: 560–599, (1984).

[33] M. Broy. Semantics of finite and infinite networks of concurrent communicating agents. *Distributed Computing*, Vol 2(1): 13–31, (1987).

[34] M. Broy and C. Lengauer. On denotational versus predicative semantics. *Journal of Computer and System Sciences*, Vol 42(1): 1–29, (1991).

[35] J.R. Burch, E.M. Clarke, D.L. Dill and L.J. Hwang. Symbolic model checking: 10^{20} states and beyond. *Proceedings of 5th IEEE Annual Symposium on Logic in Computer Science*, 428–439, IEEE Press, (1990).

[36] CCITT Blue Book, Vol X-Fasc. X.1, Recommendation Z.100, Geneva, (1989).

[37] CCITT COM X-R 26, part II.3, revised recommendation Z.100, Geneva, (1992).

[38] K.M. Chandy and J. Misra. *Parallel Program Design: A Foundation.* Addison-Wesley, (1988).

[39] A. Church. A formulation of the simple theory of types. *Journal of Symbolic Logic*, Vol 5: 56–68, (1940).

[40] A. Church and J.B. Rosser. Some properties of conversion. *Transactions of the American Mathematical Society*, Vol 39: 472–482, (1936).

[41] P. Coad, D. North and M. Mayfield. *Object Models: Strategies, Patterns and Applications.* Prentice Hall, (1995).

[42] P. Cohn. *Universal Algebra.* Reidel, (1981).

[43] J.H. Conway. *Regular Algebra and Finite Machines.* Chapman and Hall, (1971).

[44] S. Cook and J. Danies. *Designing Object Systems: Object-Oriented Modelling with Syntropy.* Prentice Hall, (1994).

[45] O.-J. Dahl, B. Myhrhaug and K. Nygaard. *SIMULA 67 Common Base Language.* Publication no. S-2, Norwegian Computer Centre, (1968).

[46] R. DeNicola and M. Hennessy. Testing equivalence for processes. *Theoretical Computer Science*, Vol 34: 83–133, (1983).

[47] W.-P. de Roever. Recursion and parameter mechanisms: an axiomatic approach. *Lecture Notes in Computer Science*, Vol 14: 34–65, (1973).

[48] W.-P. de Roever. Recursive Program Schemes: Semantics and Proof Theory. *Mathematical Centre Tracts 70*, Centre for Mathematics and Computer Science, (1976).

[49] E.W. Dijkstra. Cooperating sequential processes. In F. Genuys (ed): *Programming Languages*, 43–112, Academic Press, (1968).

[50] E.W. Dijkstra. *A Discipline of Programming.* Prentice Hall, Englewood Cliffs, NJ, (1976).

[51] E.W. Dijkstra and W.H.L. Feijen. *A Method of Programming.* Addison-Wesley, (1988).

[52] E.W. Dijkstra and C.S. Scholten. *Predicate Calculus and Program Semantics.* Texts and Monographs in Computer Science, Springer-Verlag, (1990).

[53] M. Dincbas, P. Van Hentenryck, H. Simonis, A. Aggoun, T. Graf and F. Berthier. The constraint logic programming language CHIP. *Proceedings of the International Conference on 5th Generation Computer Systems* (FGCS'88), Japan, (1988).

[54] R. Fagin, J.Y. Halpern and N. Meggido. A logic for reasoning about probabilities. *Information and Control*, Vol 87: 78–128, (1990).

[55] M. Fitting. *Intuitionistic Logic, Model Theory and Forcing.* North-Holland, (1969).

[56] M. Fitting. *First Order Logic and Automated Theorem Proving*, Second Edition. Springer-Verlag, (1996).

[57] R.W. Floyd. Assigning meanings to programs. *Proceedings of Symposia in Applied Mathematics*, Vol 19: 19–32, (1967).

[58] S. Fortune and J. Wyllie. Parallelism in random access machines. 10th ACM Symposium on the Theory of Computing, 114–118, (1978).

[59] M. Fowler and K. Scott. *UML Distilled: Applying the Standard Object Modeling Language.* Addison-Wesley, (1997).

[60] N. Francez. *Fairness.* Springer-Verlag, (1986).

[61] M. Fränzle. From continuity to discreteness — five views of embedded control hardware. Technical Report, Kiel University, (1994).

[62] M. Fränzle. A discrete model of VLSI dynamics in hybrid control application. Technical Report, Kiel University, (1994).

[63] D. Gelernter. An integrated microcomputer network for experiments in distributed programming. PhD dissertation, SUNY at Stony Brook, (1982).

[64] D. Gelernter. Generative communications in Linda. *ACM Transactions on Programming Languages and Systems*, Vol 7(1): 80–113, (1985).

[65] D. Gelernter and A. Bernstein. Distributed communications vis global buffer. *Proceedings ACM Symposium on Principles of Distributed Computing*, 0–18, (1982).

[66] J.Y. Girard. Linear logic. *Theoretical Computer Science*, Vol 50: 1–102, (1987).

[67] J. Goguen and T. Winkler. Introducing OBJ3. Technical Report SRI-CSL-88, SRI International, Computer Science Laboratory, (1988).

[68] J. Goguen, J. Thatcher, E. Wagner and J. Wright. Initial algebra semantics and continuous algebra. *Journal of the ACM*, Vol 24(1): 68–95, (1977).

[69] M.J.C. Gordon. HOL: A proof generating system for higher-order logic. In G. Birtwistle (ed): *VLSI Specification, Verification and Synthesis*, 73–128, Kluwer Academic Publishers, (1988).

[70] D. Gries. *The Science of Programming*. Springer-Verlag, (1981).

[71] R.L. Grossman, A. Nerode, A.P. Ravn and H. Rischel (eds). Hybrid Systems. *Lecture Notes in Computer Science*, Vol 776, Springer-Verlag, (1993).

[72] P. Le. Guernic, M. Le Borgne, T. Gauthier and C. Le Maire. Programming real time applications with Signal. In *Another Look at Real Time Programming*, Proceedings of the IEEE, (1991).

[73] C.A. Gunter. *Semantics of Programming Languages: Structures and Techniques*. The MIT Press, (1992).

[74] C.A. Gunter and J.C. Mitchell. *Theoretical Aspects of Object-Oriented Programming*. The MIT Press, (1994).

[75] N. Halbwachs, P. Capsi, P. Raymond and D. Polaul. The synchronous data flow programming language LUSTRE. *Proceedings of the IEEE*, Vol 79(8): 1305–1319, (1991).

[76] D. Harel. Statecharts: a visual formalism for complex systems. *Science of Computer Programming*, Vol 8: 231–247, (1987).

[77] E.C.R. Hehner. Predicative programming, Part 1 and 2. *Communications of the ACM*, Vol 27(2): 134–151, (1984).

[78] E.C.R. Hehner and C.A.R. Hoare. A more complete model of communicating processes. *Theoretical Computer Science*, Vol 26: 105–120, (1983).

[79] E.C.R. Hehner and A.J. Malton. Termination conventions and comparative semantics. *Acta Informatica*, Vol 25: 1–14, (1989).

[80] P. Henderson. *Functional Programming*. Prentice Hall, (1980).

[81] M. Hennessy. Synchronous and asynchronous experiments on processes. *Information and Control*, Vol 59: 36–83, (1983).

[82] M.C. Hennessy. *Algebraic Theory of Processes*. The MIT Press, (1988).

[83] M.C. Hennessy and R. Milner. Algebraic laws for nondeterminism and concurrency. *Journal of the ACM*, Vol 32(1): 137–161, (1985).

[84] T.A. Henzinger, Z. Manna and A. Pnueli. Timed transition systems. *Lecture Notes in Computer Science*, Vol 600: 226–251, Springer-Verlag, (1992).

[85] C.A.R. Hoare. An axiomatic basis for computer programming. *Communications of the ACM*, Vol 12: 576–583, (1969).

[86] C.A.R. Hoare. Procedures and parameters: an axiomatic approach. In *Symposium on Semantics of Programming Languages*, 102–116, Springer-Verlag, (1971).

[87] C.A.R. Hoare. Proof of correctness of data representation. *Acta Informatica*, Vol 1: 271–281, (1972).

[88] C.A.R. Hoare. *Communicating Sequential Processes.* Prentice Hall, (1985).

[89] C.A.R. Hoare. Programs are predicates. *Proceedings of International Conference on 5th Generation Computer Systems*, Tokyo, 211–218, (1992).

[90] C.A.R. Hoare and He Jifeng. The weakest prespecification. *Fundamenta Informaticae*, Vol 9: 51–84, 217–252, (1986).

[91] C.A.R. Hoare and N. Wirth. An axiomatic definition of the programming language PASCAL. *Acta Informatica*, Vol 2: 335–355, (1973).

[92] C.A.R. Hoare *et al.* Laws of programming. *Communications of the ACM*, Vol 30(8): 672–686, (1987).

[93] C.A.R. Hoare, He Jifeng and A. Sampaio. Normal form approach to compiler design. *Acta Informatica*, Vol 30: 701–739, (1993).

[94] G. J. Holzmann and D. Peled. The state of SPIN. *Lecture Notes in Computer Science*, Vol 1102: 385–389, Springer-Verlag, (1996).

[95] P. Hudak, S.L.P. Jones and P. Wadler. Report on the Programming Language Haskell, version 1.2. *ACM SIGPLAN Notices*, Vol 27(5), (1992).

[96] G. Huet. Cartesian closed categories and lambda-calculus. *Lecture Notes in Computer Science*, Vol 242, Springer-Verlag, (1986).

[97] INMOS Limited. *Occam 2 Reference Manual.* Prentice Hall, (1988).

[98] ISO 8807. *Information processing systems — Open Systems Interconnection — LOTOS — a formal description technique based on the temporal ordering of observational behaviour.* ISO standard, Geneva, Switzerland, (1987).

[99] M. Jackson. *System Development.* Prentice Hall, (1983).

[100] J. Jaffar and S. Michaylov. Methodology and implementation of a CLP system. In 4th International Conference on Logic Programming, Australia, (1987).

[101] C.B. Jones. *Systematic Software Development Using VDM.* Prentice Hall, (1986).

[102] B. Jónsson. Varieties of relation algebra. *Algebra Universalis*, Vol 15: 273–298, (1982).

[103] B. Jónsson and A. Tarski. Boolean algebra with operators, Part 1. *American Journal of Mathematics*, Vol 73: 891–939, (1951).

[104] B. Jónsson and A. Tarski. Boolean algebra with operators, Part 2. *American Journal of Mathematics*, Vol 74: 127–162, (1952).

[105] M.B. Josephs. Receptive process theory. *Acta Informatica*, Vol 29(1): 17–31, (1992).

[106] G. Kahn. The semantics of a simple language for parallel processing. In J.L. Rosenfeld (ed): *Information processing '74*, 471–475, North-Holland, (1974).

[107] G. Kahn and D. MacQueen. Coroutines and networks of parallel processes. In B. Gilchrist (ed): *Information processing '77*, 994–998, North-Holland, (1977).

[108] B. von Karger and C.A.R. Hoare. Sequential calculus. *Information Processing Letters*, Vol 53(3): 123–130, (1995).

[109] J.C. King. A program verifier. PhD thesis, Carnegie–Mellon University, (1969).

[110] S.C. Kleene. *Introduction to Meta-mathematics.* Princeton University Press, (1952).

[111] D.E. Knuth. *The Art of Computer Programming.* Addison-Wesley, (1973).

[112] L. Lamport. win and sin: Predicate transformers for concurrency. *ACM Transactions on Programming Languages and Systems*, Vol 12(3): 396–428, (1990).

[113] L. Lamport. A temporal logic of actions. *ACM Transactions on Programming Languages and Systems*, Vol 16(3): 872–923, (1994).

[114] J. Launchbury. A natural semantics for lazy evaluation. Conference Record of 20th Annual ACM SIGPLAN-SIGACTS Symposium on Principles of Programming Languages, 144–154, (1993).

[115] J.W. Lloyd. *Foundations of Logic Programming.* Springer-Verlag, (1984).

[116] S. Mac Lane. *Categories for the Working Mathematician.* Springer-Verlag, (1971).

[117] R.D. Maddux. Fundamental study Relation-algebraic semantics. *Theoretical Computer Science*, Vol 160: 1–85, (1996).

[118] Z. Manna and A. Pnueli. *The Temporal Logic of Reactive and Concurrent Systems: Specifications.* Springer-Verlag, (1991).

[119] A.J. Martin. A general proof rule for procedures in predicate transformer semantics. *Acta Informatica*, Vol 20: 301–313, (1983)

[120] P. Martin-Löf. Constructive mathematics and computer programming. In L.J. Cohen (ed): *Logic, Methodology and Philosophy of Science*, Vol 4: 153–175, North-Holland, (1982).

[121] Mathematics of Program Construction Group, Eindhoven University of Technology. Fixed-point calculus. *Information Processing Letters*, Vol 53: 131–136, (1995).

[122] J. McCarthy *et al. LISP 1.5 Programmer's Manual.* The MIT Press, (1962).

[123] W.F. McColl. BSP programming. *Proceedings of DIMACS Workshop on Specification of Parallel Algorithms*, Princeton University Press, (1994).

[124] B. Meyer. *Object-Oriented Software Construction.* Prentice Hall, (1994).

[125] R. Milner. A Calculus of Communicating Systems. *Lecture Notes in Computer Science*, Vol 92, Springer-Verlag, (1980).

[126] R. Milner. *Communication and Concurrency.* Prentice Hall, (1989).

[127] R. Milner. Polyadic π-calculus: a tutorial. Technical Report ECS-LFCS-91-180, University of Edinburgh, (1991).

[128] R. Milner, M. Tofte and R.W. Harper. *The Definition of Standard ML.* The MIT Press, (1990).

[129] C.C. Morgan. *Programming from Specifications*, Second Edition. Prentice Hall, (1994).

[130] J.M. Morris. A theoretical basis for stepwise refinement and the programming calculus. *Science of Computer Programming*, Vol 9(3): 287–306, (1987).

[131] J.M. Morris. Non-deterministic expressions and predicate transformers. *Information Processing Letters*, Vol 61: 241–246, (1997).

[132] P.D. Mosses. *Action Semantics.* Cambridge Tracts in Theoretical Computer Science, Cambridge University Press, (1992).

[133] M. Müller-Olm. Modular Compiler Verification. *Lecture Notes in Computer Science*, Vol 1283, Springer-Verlag, (1997).

[134] P. Naur. Proofs of algorithms by general snapshots. *BIT*, Vol 6: 310–316, (1969).

[135] G. Nelson. A generalisation of Dijkstra's calculus. *ACM Transactions on Programming Languages and Systems*, Vol 11(4): 517–561, (1989).

[136] Van Nguyen, A. Demers and S. Owicki. A model and temporal proof system for networks of processes. *Distributed Computing*, Vol 1(1): 7–25, (1986).

[137] M. Nivat. Nondeterministic programs: an algebraic overview. In S.H. Lavington (ed): *Information processing '80*, 17–28, North-Holland, (1980).

[138] M. Nivat and J. Reynolds (eds). *Algebraic Methods in Semantics.* Cambridge University Press, (1985).

[139] E.-R. Olderog and C.A.R. Hoare. Specification oriented semantics for communicating processes. *Acta Informatica*, Vol 23: 9–66, (1986).

[140] C.-H. Ong. Correspondence between operational and denotational semantics. In S. Abramsky, D. Gabbay and T.S.E. Maibaum (eds): *Handbook of Logics in Computer Science*, Vol 4. Oxford University Press, (1995).

[141] Open Verilog International. *Verilog Hardware Description Language Reference Manual*, Version 1.0, Open Verilog International, 15466 Los Gatos Blvd, Los Gatos, California, (1995).

[142] S.S. Owicki and D. Gries. An axiomatic proof technique for parallel programs. *Acta Informatica*, Vol 6: 319–340, (1976).

[143] S. Owre, S. Rajan, J.M. Rushby, N. Shankar and M. Srivas. PVS: combining specification, proof checking, and model checking. *Lecture Notes in Computer Science*, Vol 1102: 411–414, Springer-Verlag, (1996).

[144] D.M.R. Park. On the semantics of fair parallelism. *Lecture Notes in Computer Science*, Vol 86: 504–526, Springer-Verlag, (1979).

[145] D.L. Parnas, J. Madey and M. Iglewski. Precise documentation of well-structured programs. *IEEE Transactions on Software Engineering*, Vol 20(12): 948–976, (1994).

[146] J.L. Peterson. *Petri Net Theory and the Modeling of Systems.* Prentice Hall, (1981).

[147] C. Petri. Kommunikation mit Automaten. PhD dissertation, University of Bonn, (1962).

[148] C. Petri. Concepts of net theory. *Proceedings of the Symposium and Summer School on Mathematical Foundations of Computer Sciences*, High Tatras, 137–146, (1973).

[149] G.D. Plotkin. Call-by-name, call-by-value and the λ-calculus. *Theoretical Computer Science*, Vol 1: 125–179, (1975).

[150] G.D. Plotkin. A powerdomain construction. *SIAM Journal on Computing*, Vol 5: 452–487, (1976).

[151] G. D. Plotkin. A structural approach to operational semantics. Technical Report, DAIMI-FN-19, Aarhus University, Denmark, (1981).

[152] A. Pnueli. The temporal semantics of concurrent programs. *Theoretical Computer Science*, Vol 13: 45–60, (1981).

[153] V. Pratt. Event spaces and their linear logic. In AMAST'91: Algebraic Methodology and Software Technology, (1992).

[154] J.C. Reynolds. Definitional interpreters for higher-order programming languages. *Proceedings of the ACM Annual Conference*, 717–740, (1972).

[155] J.C. Reynolds. *The Craft of Programming.* Prentice Hall, (1981).

[156] J.C. Reynolds. The essence of Algol. In J.W. de Bakker and J.C. van Vliet (eds): *Algorithmic Languages*, 345–372, North-Holland, (1981).

[157] D.M. Ritchie and K. Thompson. The UNIX timesharing system. *Communications of the ACM*, Vol 17(7): 365–375, (1974).

[158] A.W. Roscoe. Two papers on CSP. Technical Monograph PRG-67, Oxford University Computing Laboratory, (1988).

[159] A.W. Roscoe. Model-checking CSP. In A.W. Roscoe (ed): *A Classical Mind: Essays in Honour of C.A.R. Hoare*, 353–378, Prentice Hall, (1994).

[160] A.W. Roscoe. *The Theory and Practice of Concurrency.* Prentice Hall, (1998).

[161] A.W. Roscoe and C.A.R. Hoare. Laws of Occam programming. *Theoretical Computer Science*, Vol 60: 177-229, (1988).

[162] K.L. Rosenthal. *Quantales and Their Applications.* Pitman, (1990).

[163] L. Saunders and R. Waxman. *IEEE Standard VHDL Language Reference Manual.* The Institute of Electrical and Electronics Engineers, USA, (1988).

[164] D.A. Schmidt. *Denotational Semantics.* Allyn and Bacon, (1986).

[165] D.S. Scott. Continuous lattices. *Lecture Notes in Mathematics*, Vol 274: 97–136, Springer-Verlag, (1972).

[166] D.S. Scott. Data types as lattices. *SIAM Journal on Computing*, Vol 5: 522–587, (1976).

[167] D.S. Scott. Domains for denotational semantics. *Lecture Notes in Computer Science*, Vol 140: 577–613, Springer-Verlag, (1982).

[168] D.S. Scott and Christopher Strachey. Towards a mathematical semantics for computer languages. *Proceedings of 21st Symposium on Computers and Automata*, 19–46, Polytechnic Institute of Brooklyn, (1971).

[169] M.B. Smyth. Power domains. *Journal of Computer and System Science*, Vol 16(1): 23–26, (1978).

[170] M.B. Smyth and G.D. Plotkin. The category-theoretic solution of recursive domain equations. *SIAM Journal on Computing*, Vol 11: 761–783, (1982).

[171] J.M. Spivey. *The Z Notation: A Reference Manual*, Second Edition. Prentice Hall, (1992).

[172] J. Stoy. *Denotational Semantics.* The MIT Press, (1977).

[173] C. Strachey. Fundamental concepts in programming languages. Unpublished lecture notes, International Summer School in Computer Programming, Copenhagen, (1967).

[174] A. Tarski. On the calculus of relations. *Journal of Symbolic Logic*, Vol 6(3): 73–89, (1941).

[175] A. Tarski. A lattice-theoretical fixpoint theorem and its applications. *Pacific Journal of Mathematics*, Vol 5: 285–309, (1955).

[176] R.D. Tennent. The denotational semantics of programming languages. *Communications of the ACM*, Vol 19: 437–453, (1976).

[177] R.D. Tennent. *Semantics of Programming Languages.* Prentice Hall, (1991).

[178] R.D. Tennent. Correctness of data representation in Algol-like languages. In A.W. Roscoe (ed): *A Classical Mind: Essays in Honour of C.A.R. Hoare*, 405–418, Prentice Hall, (1994).

[179] A.M. Turing. Checking a large routine. Report of a Conference on High Speed Automatic Calculating Machines, 67–69, University Mathematical Laboratory, Cambridge, (1949).

[180] D.A. Turner. An overview of Miranda. In D.A. Turner (ed): *Research Topics in Functional Programming*, Addison-Wesley, (1990).

[181] J.T. Udding. Classification and composition of delay-insensitive circuits. PhD dissertation, Eindhoven University of Technology, (1984).

[182] J.D. Ullman. *Elements of ML programming.* Prentice Hall, (1994).

[183] L.G. Valiant. A bridging model for parallel computation. *Communications of the ACM*, Vol 33(8): 103–111, (1990).

[184] G. Winskel. *The Formal Semantics of Programming Languages.* The MIT Press, (1993).

[185] N Wirth and C.A.R. Hoare. A contribution to the development of ALGOL. *Communications of the ACM*, Vol 9: 413–431, (1966).

[186] S. Wolfram. *The Mathematica Book.* Cambridge University Press, (1996).

[187] Zhou Chaochen, C.A.R. Hoare and A.P. Ravn. A calculus of durations. *Information Processing Letters*, Vol 40(5): 269–276, (1992).

[188] J. Zwiers. Compositionality, Concurrency and Partial Correctness. *Lecture Notes in Computer Science*, Vol 321, Springer-Verlag, (1989).

Index