J. VAN LEEUWEN (Ed.), Handbook of Theoretical Computer Science. Volume B – Formal Models and Semantics. Amsterdam: Elsevier – Cambridge, Mass.: The MIT Press, 1990. 1273 p. prijs *f* 325.00 ISBN 0–444–88074–7 [*f* 550.00 for both volumes ISBN 0–444–88075–5].

Whereas Volume A of this handbook (*Meded. Wisk. Gen.* **36** (1993) 268–272 / B170–174) has been devoted to a single subject (complexity theory), this Volume B is more diverse. Roughly, we can distinguish the following areas: formal languages and rewriting (Chapters 1–6), programming (7–10), semantics (11–13), logics (14–16), databases (17), and concurrency (18–19). The chapters in this volume are:

1. *Finite Automata* (D. Perrin, pp. 1–57) is a survey of regular languages: an algebraic introduction, syntactic monoids, regular expressions, ambiguity, star-height, finite power property, Kleene's theorem and pumping lemma. Then a few topics are treated in more depth: some subregular language families (star-free, aperiodic, locally testable, piecewise testable languages), automata for pattern matching, recognizable sets of numbers, random access automata, and Schützenberger's Theorem.

2. *Context-Free Languages* (J. Berstel & L. Boasson, pp. 59–102) also presents some less traditional results: viz. the Hotz group, several pumping lemmas, interchange lemma and square-free words, and degenerated iterative pairs. The second half of this paper is devoted to nongenerators of the family of context-free languages, and to context-free groups. Nongenerators are considered with respect to rational cones (algebras of which the carrier set is a language family and the operations are finite-state transductions). A finitely generated group is context-free if the language of all words equivalent to the unit is context-free. Global and local characterizations of these groups are given.

3. *Formal Languages and Power Series* (A. Salomaa, pp. 103–132) starts to address: grammars and automata, Post systems, Markov algorithms, Lindenmayer systems, Thue sequence, equality sets, and a proof of Ehrenfeucht's conjecture. The second part is devoted to the relation between languages and formal power series in noncommuting variables. Regular languages can be characterized in terms of certain rational power series as context-free languages can by some algebraic power series. In this context (un)decidability results, the commutative case, and ambiguity are discussed.

4. *Automata on Infinite Objects* (W. Thomas, pp. 133–191) deals with finite-state acceptors operating on infinite words or trees. In part I ω-*regular languages* are characterized algebraically, logically, and by automata. Closure under Boolean operations and topological issues are treated as well. Special topics, like star-free ω-languages, their relation with propositional linear-time logic, and context-free ω-languages conclude this part. Part II is devoted to finite top-down automata for rooted node-labeled infinite trees, characterizations of the corresponding families of tree languages, decidability of the emptiness problem, the relation to logics and to regular infinite games.

5. *Graph Rewriting: An Algebraic and Logic Approach* (B. Courcelle, pp. 193–242) uses logics, universal algebra, and category theory to describe properties of (hyper)graphs. Context-free sets of hypergraphs are defined by hyperedge-replacement grammars or as a solution of an appropriate set of equations. Then the connections to monadic second-order logic are considered. Other issues include defining sets of graphs by forbidden minors, recognizable sets of graphs, complexity, and sets of infinite hypergraphs.

6. *Rewrite Systems* (N. Dershowitz & J.P. Jouannaud, pp. 243–320) is an overview of sets of rules used to rewrite repeatedly until the simplest possible form is obtained. After introducing syntactic notions attention is focused on initial algebra semantics, the Church-Rosser property, confluence, coherence, termination, and satisfiability (by syntactic or semantic unification, or by "narrowing"). The last part of this paper deals with critical pairs and completion (cf. the Knuth-Bendix algorithm) and variants of rewrite systems (order-sorted, conditional, priority, and graph rewriting).

7. *Functional Programming and Lambda Calculus* (H.P. Barendregt, pp. 321–363) first provides the rudiments of λ-calculus, combinatory logic and their relation to computability. Then semantics of the λ-calculus are treated: operational (reduction, Church-Rosser

property, normal form, reduction strategy) and denotational (set-theoretic models due to Scott). Two ways of extending the λ-calculus are considered: the λδ-calculus, and various typed λ-calculi with their derivation systems. This chapter concludes with combinatory systems, and implementation issues of functional languages.

8. *Type Systems for Programming Languages* (J.C. Mitchell, pp. 365–458) studies typed λ-calculi in more detail: context-sensitive syntax and defining languages by typing rules, an equational proof system and reduction rules, normal form, Church-Rosser property, and semantics of typed λ-calculi. The next topic is the notion of logical relation, i.e. a family $\{R^\sigma\}$ of typed relations such that $R^{\sigma\to\tau}$ for type $\sigma\to\tau$ is determined by R^σ and R^τ in a way that guarantees closure under application and λ-abstraction. The final part of the paper surveys type systems with polymorphism, data abstraction, and with algorithms for automatically inferring type information.

9. *Recursive Applicative Program Schemes* (B. Courcelle, pp. 459–492) investigates properties of recursive programs with emphasis on control structures rather than data types. After defining syntax and semantics (operational semantics in discrete and in continuous interpretations, computation rules, least fixed-point semantics) of these program schemes the paper deals with two topics: classes of interpretations and transformations of program schemes.

10. *Logic Programming* (K.R. Apt, pp. 493–574) provides an overview of theoretical issues related to PROLOG. The central theme is a form of theorem proving called *SLD-resolution* that serves as computation rule for logic programs. Semantics is introduced in order to establish soundness of SLD-resolution as well as some forms of its completeness. Next it is shown that all recursive functions can be computed by logic programs. Since SLD-resolution only allows the derivation of positive statements, the second half of the paper deals with the derivability of negative statements: the infectiveness of the "closed world assumption", the "negation as (finite) failure" rule, and completion of a program. Other topics (general programs, deductive databases, integration of logical and functional programming, applications in artificial intelligence) are also discussed.

11. *Denotational Semantics* (P.D. Mosses, pp. 575–631) is a lucid introduction to models for programming languages. It discusses formalisms used in denotational semantics: concrete and abstract syntax, semantic functions and the properties of compositionality and full abstractness, and semantic domains. Then it shows standard techniques that are applied in the denotational description of fundamental concepts (like sequential computation, scope rules, local variables, guarded commands, interleaving) in programming: environment, store, and continuation (i.e. the remainder of the program).

12. *Semantic Domains* (C.A. Gunter & D.S. Scott, pp. 633–674) has a more formal and restricted character; it is devoted to a single aspect of denotational semantics. Complete partial orders (cpo's), continuous functions on cpo's, and their fixed points are treated to provide semantics for recursively defined functions. Domains are algebraic cpo's with a countable number of compact elements. The subset of effectively presented domains is a suitable basis for defining computable functions. Then operators on domains are discussed with emphasis on (convex) powerdomains and bifinite domains.

13. *Algebraic Specification* (M. Wirsing, pp. 675–788) provides techniques, based on many-sorted algebras, for data abstraction, and the structured specification, validation and analysis of data structures. After defining abstract data types algebraically, attention is focused on semantics (initial, loose and terminal semantics), four other semantical concepts (error, partial, continuous and order-sorted algebra), and algebras for specifying nondeterminism and concurrency. The main other topics in this long chapter are: hidden symbols, constructors for hierarchical and structured specification, institutions, observable behavior, parameterized specification, and specification languages.

14. *Logics of Programs* (D. Kozen & J. Tiuryn, pp. 789–840) are formal systems for specifications, proving that specifications are met by a program, determining equivalence of programs, comparing the expressive power of programming constructs, etc. These issues are studied in the framework of *dynamic logic* (DL), a modal logic in which a program p corresponds to a modal operator $<p>$. For a formula ϕ such an operator yields a

modal construct $<p>\phi$ which states: "It is possible to execute p and halt in a state satisfying ϕ". The emphasis is on propositional DL and first-order DL; the latter one eventually extended with arrays, stacks or nondeterministic assignments. Other approaches (nonstandard DL, algorithmic logic, temporal logic) are briefly mentioned.

15. *Methods and Logics for Proving Programs* (P. Cousot, pp. 841–993) introduces a simple language of while-programs: syntax, its operational, relational and axiomatic semantics (Hoare logic or HL), and the notion of partial correctness. A proof method for establishing partial correctness is given and its soundness and completeness are shown. This method is generalized to include termination, total correctness and liveness. A study of HL follows: semantical and syntactical point of view, provability, (in)completeness, Cook's and Clarke's expressiveness. Finally, the effect on HL of extending the language with data structures, blocks, procedures, goto's, and parallelism is considered.

16. *Temporal and Modal Logic* (E.A. Emerson, pp. 995–1072) is a counterpart to Chapter 15: viz. it deals with reasoning about nonterminating concurrent programs such as operating systems and network protocols. The time-varying behavior of these programs is modeled by modalities as *sometimes* and *always*. Time can be either discrete or continuous, linear or branching (to model nondeterminism), emphasis may be on intervals rather than on points. The resulting propositional and first-order logics are studied: syntax, semantics, decidability, complexity, axiomatizability, and expressiveness. Applications of temporal logic to program reasoning (correctness, verification, mechanical synthesis from temporal logic specifications) are discussed as well.

17. *Elements of Relational Database Theory* (P.C. Kanellakis, pp. 1073–1156) covers basic aspects of the relational data model: its specification (database scheme: layout of the tables and the functional and other dependencies) and its operation (relational algebra queries, relational calculus). In the second part emphasis is on dependencies (classification, decidability and complexity bounds on implication problems), queries (computable queries, fixpoint queries, relation to logic programming, deductive databases, query optimization), problems of incomplete information, and database updates.

18. *Distributed Computing: Models and Methods* (L. Lamport & N. Lynch, pp. 1157–1199) is a clear overview of process models – i.e. describing the concurrent execution of a number of sequential processes – mainly of models that use message passing as mechanism for interprocess communication. General notions (verification, specification, mutual exclusion, critical section, safety and liveness) are covered as well as some typical distributed algorithms (shared variable algorithms, distributed consensus, network algorithms, and concurrency control in databases).

19. *Operational and Algebraic Semantics of Concurrent Processes* (R. Milner, pp. 1201–1242) is a complementary, more formal (algebraic) treatment of concurrency. The main part of this chapter is devoted to the author's Calculus of Communicating Systems: its syntax and operational semantics (labeled transition systems). The central theme in this paper is observational equivalence of processes. The discussion concentrates on the algebra of equality: equivalence and congruence relations, and the existence and uniqueness of solutions of equations which define processes recursively.

The numbers of references in this volume range from 24 (Chapter 12) to 410 (Chapter 15) with an average of 135. Of course, the short descriptions in this review do not justice to the wealth of results in each chapter. There remains the obvious question whether some subjects are missing. Personally I missed subjects like parsing and compiling, a few chapters on "less traditional ways of computing" (neural networks, cellular automata, genetic algorithms, relation to dynamical systems), and some more attention to process algebras and structural operational semantics – although there already is a lot of semantics in Volume B. Anyway, the editor deserves credit for his enormous job in bringing these valuable surveys together.

P.R.J. Asveld