ALGOL-like Languages

edited by Peter W. O'Hearn and Robert D. Tennent

Birkhäuser Boston, 1997, 286 p. + 348 p.

Progress in Theoretical Computer Science

ISBN 0-8176-3880-6 (Volume 1)

ISBN 0-8176-3937-3 (Volume 2)

This collection aims at making "the most significant material on ALGOL-like languages conveniently available to graduate students and researchers" because "in recent years there has been a remarkable convergence of interest in programming languages based on ALGOL-60".

The paradox of ALGOL-60 - a language never widely used, but never forgotten, and still thought over as source, root or foundation for many modern languages or programming paradigms - was probably most remarkably stated by Hoare (also quoted in the Introduction): "a language so far ahead of its time that it was not only an improvement on its predecessors but also on nearly all its successors". Indeed, many procedural, object-oriented or (purely) functional languages are influenced by concepts identified as "ALGOL-like" (e.g. a procedure mechanism based on the fully typed, call-by-name lambda calculus; not assignable procedures, variables and other denotable meanings; purely "mathematical" expressions without side-effects).

Fortunately, the genuine purpose, approach, orbit and contents of this substantial collection are asserted laconically on its cover - and it is certainly no coincidence that this cover was designed by Robert Tennent, one of the editors, author/co-author of four of the eleven papers included in the second volume: "λ", as an emblem for the lambda calculus, groundwork for all functional programming languages, is intertwined - according to the principles of the ancient Chinese philosophy - with ":=", as an emblem for the assignment statement, keystone of any imperative language (the only unanswered question: has the white colour for "λ", respectively the black one for ":=", any significance?).

Thus, after a comprehensive introduction, the first volume is divided into three parts and the second one into four; each part comprises between two and five Chapters usually consisting of reprinted or revised papers.

The **first Part**, *Historical Background*, has two Chapters: *Revised Report on the Algorithmic Language ALGOL-60* (authored by such legendary personages like Naur, Backus, McCarthy, van Wijngaarden ...), giving a complete description of the language, and *The Varieties of Programming Language* (Ch. Strachey), suggesting an analysis of the domains used in programming languages - in fact an introductory paper to denotational semantics.

The **second Part**, *Basic Principles*, reprints five papers. *The Essence of ALGOL* (J. C. Reynolds), taking into account that "although ALGOL-60 has been uniquely influential in programming language design its descendants have been significantly different than their prototype", states the principles which - Reynolds believes - embody the essence of ALGOL, describes a model that satisfies these principles, and illustrates this model with a language, IDEALIZED ALGOL, that, while more uniform and general, retains the character of ALGOL. **Chapters 4 to 7** formalise or develop some of the key ideas in this paper. So, P. W. O'Hearn in *ALGOL and Functional Programming* shows that in IDEALIZED ALGOL observational equivalence conservatively extends it in its assignment-free functional sub-language (a technical result further exemplifies its functional character). To formalise ALGOL's notion of "orthogonality" (i.e. ALGOL-60 "orthogonally" extends a simple imperative programming language with a typed λ-calculus), in *Orthogonality of Assignments and Procedures in ALGOL*, M. Felleisen and S. Weeks define an extended λ-calculus that models ALGOL, and show that the proofs of the Church-Rosser and Strong Normalisation theorems are combinations of separate theorems for each sub-language. **Chapter 6** presents, in another paper of J. C. Reynolds's, *IDEALIZED ALGOL and its Specification Logic*, several innovations: specifications as predicates of the environments; logical connectives at the level of specification

formulas - i.e. treating Hoare's triples as atomic formulas in a first-order theory; "non-interference" predicates to deal with aliasing; call-by-name as the basic parameter mechanism. In *Towards Fully Abstract Semantics for Local Variables: Preliminary Report*, A. R. Meyer and K. Sieber demonstrate that traditional marked-store models fail to validate some extremely simple expected equivalencies and show the weaknesses of existing program logics for imperative languages with procedures, which were sound in such models.

The **third Part**, *Language Design*, starts with the *Design of the Programming Language FORSYTHE* (J.C.Reynolds), "a descendant of ALGOL-60 intended to be as uniform and general as possible, while retaining the basic character of its progenitor"; the innovation: "intersection" (conjunctive) types that simplify, unify and generalise the type structure. **Chapter 9**, *Assignments for Applicative Languages* (V. Swarup, U. S. Reddy, and E. Ireland), is an approach to combining imperative and functional aspects by using types to separate imperative and applicative parts of a program. The last Chapter of the first Volume, *Syntactic Control of Interference* (again, J. C. Reynolds), demonstrates the possibility of avoiding aliasing and covert interference by imposing syntactic restrictions, without prohibiting the kind of constructive interference which occurs with high-order procedures or SIMULA classes.

In the second Volume, ALGOL-like languages are studied from various modern denotational semantics points of view.

The **fourth Part**, *Functor- Category Semantics*, consists of two papers. In *Functor Categories and Store Shapes*, F. J. Oles outlines how functor categories can be used to explain the semantics of ALGOL-like languages having rich type structure, higher-order procedures (with arbitrarily complex types), imperative features and block structure. In *Using Functor Categories to Generate Intermediate Code*, J. C. Reynolds defines the translation of an ALGOL-like language in a uniform way that avoids unnecessary temporary variables, provides control-flow translation of Boolean expressions, permits on-line expansion, and minimises storage overhead.

The **fifth Part**, *Specification Logic*, also comprises two Chapters. In *Semantical Analysis of Specification Logic*, R. D. Tennent adapts the functor category approach to allow the treatment of certain "intensional" properties of commands within a formally extensional intuitionistic first-order theory. In *Semantical Analysis of Specification Logic 2*, P. W. O'Hearn and R. D. Tennent present a new semantic interpretation of non-interference (for phrases of arbitrary type) which is equivalent to the interpretation given earlier to phrases of basic type.

The **sixth Part**, *Procedures and Local Variables*, is composed of three Chapters. In *Full Abstraction for the Second-Order Subset of an ALGOL-like Language*, K. Sieber deals with a denotational semantics constituting the first full-abstraction result for a block-structured language with local variables. In *Parametricity and Local Variables*, P. W. O'Hearn and R. D. Tennent also use logical relations as a support for a conceptual connection with representation independence and data abstraction. **Chapter 17** reprints *Reasoning About Local Variables with Operationally-Based Logical Relations* (A. M. Pitts), a different kind of paper, because it concentrates on operational rather than on denotational semantics; it separates the reasoning principles, which are implicit in the models based on logical relations, from their use in constructing the models.

The **seventh** - and last - **Part**, *Interference, Irreversibility and Concurrency*, has four Chapters. In *Syntactic Control of Interference Revisited*, P. W. O'Hearn, A. D. Power, M. Takeyama and R. D. Tennent describe a new type system (and verify that reduction preserves typings) and define a class of "bireflective" models providing a sound interpretation of the type system. **Chapter 19** reprints *Global State Considered Unnecessary: Introduction to Object-Based Semantics* (U. S. Reddy), a prominent paper that proposes a novel approach to the semantics of imperative programming languages, based on a notion of objects, and characterises them in terms of their observable behaviour; states are regarded as part of the internal structure of objects and play no role in the observable behaviour (this leads to considerable accuracy in the semantic modelling of locality and single-threadedness properties of objects). In *Linearity, Sharing and State: A Fully Abstract Game Semantics for IDEALIZED ALGOL with Active Expressions*, S. Abramsky and G. McCusker present a semantics using games, quite unlike traditional denotational models of state; their model takes into account the irreversibility of changes in state and makes the difference between copying

and sharing of entities explicit. In the last Chapter, *The Essence of PARALLEL ALGOL*, S. Brookes formulates a functor-category model for a parallel version of IDEALIZED ALGOL, adapting the possible-worlds model to the parallel setting, and generalising the "transition traces" model to the procedural setting.

Of course, regarding the proportions, scope, level, and contents of such an ensemble - as well as the very nature of a collection - some redundancy and heterogeneity are unavoidable (but not necessarily inappropriate); in fact, the editors manage very well the chore of "taskbuilding" by way of editors' notes (e.g. on page 81, such a note qualifies a statement as "slightly inaccurate", explains the error and points to another chapter) and "pointers" in the Reference sections.

The first impression, namely that the topics attract only those interested in theoretical Computer Science (or just in its history: next very recent papers, there are some which had been written about thirty years ago), may be deluding if observing that the areas covered include program robustness, concurrency, run-time efficiency or even pertinent aspects of modern programming paradigms.

On the other hand, the allegation that in the first Volume "all of the material should be accessible to beginning graduate students in programming languages and theoretical Computer Science" is rather hard to defend: at least some background knowledge in Logic (e.g. possible-worlds semantics) and domain theory (e.g. Scott domains) were as useful for Chapter 9 as category theory (e.g. functor-categories) for Chapter 8. For the second Volume, in addition to the fields referred to in the Introduction, some background in Intuitionistic Logic could help.

To summarise, this extensive and carefully balanced collection carries much more information for pragmatic people than a hasty reader might guess from its first pages, or from the rather unpromising title.

The reviewer is pleased to remark the qualitative and prolific editorial work of Birkhäuser Verlag, and their gentle and convincing offer of most interesting titles which a reviewer can only be happy with.

Boldur Barbat

Dr. **Boldur Barbat** was born in Resita, Romania in 1938. He graduated in Electronic Engineering from the Polytechnical Institute of Bucarest in 1959 and obtained his Ph.D in Computer Science in 1995. Currently he is senior programmer at the Research Institute for Informatics in Bucharest, and Visiting Professor at the "Lucian Blaga" University, the Faculty of Engineering, Sibiu, Romania. He has authored / co-authored several books. His papers were published in national and international journals and presented at national and international conferences. His research interests are in real-time programming, expert systems, uncertain knowledge processing.