cussed, with emphasis on arithmetic expression code generation. This chapter includes a discussion of fundamental code-block optimizations, including redundant store elimination and common subexpression finding, with a following chapter discussing straight-line register allocation and temporary storage minimization in more detail. The book ends with a quick comparative survey of various compiler-writing systems.

J. T. SCHWARTZ

Courant Institute of Mathematical Sciences
New York University
New York, New York 10012

46[12].—P. J. KIVIAT, R. VILLANEUVA & H. M. MARKOWITZ, *The SIMSCRIPT* II *Programming Language*, Prentice-Hall, Inc., Englewood Cliffs, N. J., 1969, xiii + 386 pp., 25 cm. Price $10.95 cloth, $6.95 paper.

SIMSCRIPT is a programming language whose primary orientation is towards the programming of computer simulations, but which has the facilities of a general-purpose language. The authors of this book have chosen to emphasize the general-purpose aspects of SIMSCRIPT rather than its simulation capabilities.

Stylistically, SIMSCRIPT is a very "smooth" language, and its design is highly professional. The syntax, like that of COBOL, is intended to make programs read like English sentences, though briefer modes of expression are permitted. With the syntax stripped away, the algebraic part of the language would look like FORTRAN with a few ALGOL features, such as conditional statements and DO loops with variable parameters. There are additional facilities for text manipulation and some rather elaborate report-generating capabilities (quite useful, of course, in simulation experiments). The input-output is well planned and easy to use.

It is doubtful that SIMSCRIPT would attract many users, however, solely on the basis of its general-purpose facilities. The strength of the language lies in its capabilities for handling entities, sets and attributes. An *entity* is a computational object capable of having attributes, of belonging to sets, and of owning sets; an owned set may be thought of as a set-valued attribute. The attribute facility may be used to create PL/I-like structures, though the set operations have no immediate PL/I counterpart. Entities may be removed from or added to sets in a number of different ways, corresponding to various forms of queueing.

The simulation facilities are based upon the notions of a system clock, which keeps track of simulated time, and of events which are computations to be carried out at a certain point in simulated time. Events may arise either endogonously (internally generated) or exogonously (externally generated). After all events associated with the current time are executed, the system clock is simply advanced to the time of the next scheduled event or events. Execution of an event, of course, can cause the creation of new events, which may be scheduled at the current time or at later times.

I disagree with the authors' claim that the general-purpose part of SIMSCRIPT is comparable in power with ALGOL or PL/I. For instance, SIMSCRIPT does not have the ALGOL block structure, though it does permit recursive functions. It also lacks certain conveniences such as the ability to start array subscripts at values other

than one. The character string variables are of two types: ALPHA variables, which fit into a single computer word, and TEXT variables, which are arbitrarily long strings. The distinction between the two is thus implementation-based, and could be confusing to the neophyte.

The implementation of SIMSCRIPT uses dynamic storage allocation for entities, and for arrays as well. The result has a great deal of flexibility, though I suspect some of it at the price of efficiency. The SIMSCRIPT "DO" statement, like the ALGOL "FOR" statement, permits the parameters of the "DO" to vary as the loop is executed; thus, unless the compiler is very clever about it, execution of "DO" loops will involve a great deal of unnecessary recomputation.

The book is written in five sections, in order of increasing difficulty. They are described by the authors as:

1. A simple teaching language designed for nonprogrammers.
2. A language comparable in power to FORTRAN.
3. A language comparable in power to ALGOL or PL/I.
4. The entity-attribute-set features of SIMSCRIPT.
5. The simulation-oriented part of SIMSCRIPT.

This arrangement was made for pedagogical reasons, but it does not quite succeed. An experienced programmer will find the book slow reading if he tries to master all the details; a novice would have a great deal of difficulty. However, the information is all there, and a determined reader will assimilate it sooner or later.

I would recommend this book for those interested in the field of programming languages, for those who need to write a computer simulation and are shopping around for a language in which to write it, and for those who are working on problems where the set and property manipulation facilities of SIMSCRIPT would be helpful.

PAUL ABRAHAMS

Courant Institute of Mathematical Sciences
New York University
New York, New York 10012

47[12].—JEAN E. SAMMET, *Programming Languages: History and Fundamentals*, Prentice-Hall, Inc., Englewood Cliffs, N. J., 1969, xxx + 785 pp., 24 cm. Price $18.00 ($13.50 student edition).

Programming Languages: History and Fundamentals is a monumental and encyclopedic treatment of its field. The primary aim of the book, as stated by the author, is to provide, ". . . in one place, and in a consistent fashion, fundamental information on programming languages, including history, general characteristics, similarities, and differences." This aim has without doubt been achieved, and achieved well. A second aim is ". . . to provide specific basic information on all the significant, and most of the minor, higher level languages developed in the United States." This aim also has been achieved; there is an impressive collection of about 120 languages described in varying amounts of detail.

The book is organized into three introductory chapters, six chapters on languages grouped more or less according to application areas, and two chapters on unimple-