```
apply(function, listvarg, alist)
  if function = Ident
  then exec(Ident, Listarg, alist)
  else co function = (LAMBDA S1. S2) co
    eval(S2, pairlis(S1, listarg, alist))
  fi
evlis(Listarg, alist)
  if Listarg = NIL
  then NIL
  else co Listarg = (A1 Listarg') co
    (eval(A1, alist). evlis(Listarg', alist))
  fi
evcon(Listcond, alist)
  if Listcond = NIL
  then NIL
  else co Listcond = ((S1 S2) Listcond')) co
    if eval(S1, alist) ≠ NIL
    then eval(S2, alist)
    else evcon(Listcond', alist)
    fi
  fi
pairlis(listvar, listvarg, alist)
  if listvar = NIL
  then alist
  else co Listvar = (V. LV), Listvarg = (VA. LVA) co
                ((V . VA) . pairlis(LV, LVA, alist))
  fi
```

Appendix 4 Example of object code

Function FACT (Factorial)
Source Language:

```
(LAMBDA(N) (COND((EQUAL N 0)1)
                 (T(TIMES N(FACT(SUB1 N))))))
```

Object code;

```
1  : PUSHB 'N 0 ;          12 : BRN 22 ;
2  : PUSHV 'N 0 ;          13 : PUSHV 'N 0 ;
3  : PUSHQ '0 ;            14 : PUSHV 'N 1 ;
4  : LINK 2 EQUAL ;        15 : LINK 1 SUB1 ;
5  : POP 2 ;               16 : POP 1 ;
6  : BRN 10 ;              17 : LINK 1 FACT ;
7  : PUSHQ '1 ;            18 : POP 1 ;
8  : POP 1 ;               19 : LINK 2 TIMES ;
9  : BR 21 ;               20 : POP 2 ;
10 : POP ;                 21 : POP 1 ;
11 : PUSHQ 'T ;            22 : POPB 'N
                           23 : RETURN
```

References

DERANSART, P. (1977). Definition and implementation of a Lisp system, using semantic attributes, *5th annual III Conference.*

DERANSART, P. (1978a). Technique de preuve par attributs appliquée à un compilateur Lisp, Coopération Franco-Soviétique, Paris.

DERANSART P. (1978b). Proof and Synthesis of Semantic Attributes in compiler definition, IRIA—LABORIA report no. 333.

HOARE, C. A. R. (1969). An Axiomatic Basis for Computer Programming, *CACM*, Vol. 12 no. 10.

LONDON, R. L. (1971). Correctness of two compilers for a Lisp subset. Repot n° CS 240, University of Stanford, October 1971.

LORHO, B. (1974). De la définition à la traduction des langages de programmation: méthode des attributs sémantiques, Thèse, Université Paul Sabatier de Toulouse, 29 Novembre 1974 (French).

LUX, A. (1975). Etude d'un modèle abstrait pour une machine Lisp et de son implémentation, Thèse—Université Scientifique et Médicale de Grenoble, 19 mars 1975 (French).

PAIR, C., AMIRCHAHY, M. and NEEL D. (1976). Preuve de descriptions de traitement de textes par Attributs, IRIA-LABORIA (French).

Book reviews

Current Trends in Programming Methodology, Vol III Software Modeling edited by R. Jeh and K. M. Chandy, 1978; 379 pages. (*Prentice-Hall*, £13·85)

Mathematical modelling (often less aptly named operations research) has been widely used as an aid to management in improving the efficiency of a business. It is now recognised that the varied techniques which have evolved can also be applied to the design and tuning of computer systems, often noted for their misuse of resources. This volume, with an odd title, attempts to describe the main methods of modelling and apply them to a range of problems associated with computer performance. The individual authors of the nine chapters are able to write with authority.

The major topics discussed are statistical analysis of performance data; the analysis of a network of queues using simulation, theory and a comprehensive package called RESQ (Research Queuing Analyzer); graph theory applied to parallel computation, frequency monitoring, job scheduling; deadlocks; memory management; mathematical programming.

Each chapter is well supplied with examples relating to computer performance.

The book should be of value to the traditional system designer seeking additional quantitative support in decision-making; it should also interest the OR worker who is seeking new challenging problems. Finally, all undergraduates in computer science should be exposed to an introduction to the topics discussed in the book. It is well produced and has an extensive bibliography of 313 references; surprisingly, there is no index.

T. VICKERS (Twickenham)

Introduction to Formal Language Theory, by M. A. Harrison, 1978; 594 pages. (*Addison-Wesley*, £15·75)

This book covers most of the material one would expect from the title. The author says he assumes knowledge of finite automata, Turing machines and computer programming; I feel maths is a more important prerequisite. The approach taken is relatively formal, proofs of the major theorems (constructions) are given but some proofs are left to the reader.

The first half of the book covers type 2 and 3 languages/grammars and their associated automata including some discussion on ambiguity and decision problems. A brief look at the rest of the Chomsky hierarchy and some representation theorems then lead to the meat of the remaining half of the book. This is one of the best accounts I have seen of deterministic languages and parsing problems.

The material as presented develops in a logical and coherent way but I would have preferred to have seen more explanation for the motivation of the study. My major criticism of the book is that the computer science professional (or indeed student) could work all the way through it yet still validly ask the question 'why bother?'. There are many good reasons for studying this subject and I wish the author had brought these out.

The book is well printed with good examples, problems and illustrations. There are commendably few printing errors for a book of this complexity.

In summary, I find this book sound yet unexceptional; it is neither inspired nor bad.

D. SIMPSON (Sheffield)