

- tial-match Search in BD trees*. Technical Report 86-5, Department of Computational Science, University of Saskatchewan, Saskatoon (1986).
8. D. E. Knuth, *The Art of Computer Programming*. Vol 3: *Sorting and Searching*. Addison-Wesley, Reading, Mass. (1973).
 9. Y. Ohsawa and M. Sakauchi, The *BD-tree* – a new *N*-dimensional data structure with highly efficient dynamic characteristics. *Proceedings of the IFIP Conference, Paris*, pp. 539–544 (1983).
 10. J. Orenstein, Multidimensional tries used for associative searching. *Information Processing Letters* 14 (4), 150–157 (1982).
 11. R. Rivest, Partial-match retrieval algorithms. *SIAM Journal on Computing* 5 (1), 19–50 (1976).

Book Reviews

H. BUNKE AND A. SANFELIU (eds)
Syntactic and Structural Pattern Recognition Theory and Applications
 ISBN 9971-50-566-5; World Scientific Publishing Co., Pte Ltd, Singapore. Hardcover £52.00 Softcover £32.00

The book is a collection of 18 chapters, written by different authors, which together form an overview of a wide range of methods for syntactic and structural pattern recognition. The first twelve chapters describe the theory, and the last six chapters describe the applications. The chapters are for the most part well written, and they are coherent in style and presentation. The level of the presentation is advanced; it is suitable for those already acquainted with the field, or with a good background in computer science.

The aim of pattern recognition is to devise efficient and reliable methods for recognising an object from one or more images. Recognition can be difficult for a number of reasons. The images may be distorted by noise, parts of the object may be concealed from view, or the object may be intrinsically difficult to describe. The method in structural pattern recognition is to first find the images of the parts out of which the object is composed, and then to work out the way in which the parts fit together to make the complete object. The syntactic approach to structural pattern recognition is based on an analogy with the formal languages used in computer science. The object is described by a string of symbols. The individual symbols correspond to parts of the object, and the arrangement of symbols in the string is related to the arrangement of parts to form the object. Not all arrangements of symbols are acceptable as representations of the object. The legitimate arrangements are generated by a grammar, which is usually regular or context free. The problem of recognising an object becomes, at least in part, the problem of deciding whether a particular string of symbols can be generated by a given grammar. If the structure of the object cannot be described easily by a string of symbols then a tree, graph, or array of symbols can be used instead.

There are a number of ways of elaborating the basic formal grammar approach to syntactic pattern recognition. For example, the grammars can be given attributes. Each symbol occurring in the derivation of a string has an attribute, which is usually a numerical value. The attributes are calculated according to a set of rules incorporated into the grammar. There are also hybrid methods which combine structural methods with statistical methods, or with methods from artificial intelligence such as formal logic, or semantic nets.

The introductory chapters give a good but condensed description of the formal language theory underlying the syntactic approach to structural pattern recognition. Classes of

grammars for generating strings, trees, graphs and arrays are described, together with parsing algorithms for deciding whether a particular structure is generated by a given grammar. It is shown how to attach probabilities to the production rules of a grammar in order to create a stochastic grammar which generates structures with particular probabilities. There follow chapters on matching, grammatical inference, hybrid pattern recognition methods, and industrial applications.

Some of the gaps between the theory and the practice of syntactic pattern recognition are described in two excellent chapters, by L. Miclet and H. S. Baird, respectively. Miclet discusses the key problem of grammatical inference. In applications of syntactic pattern recognition it is necessary to infer a grammar. Ideally this should be done automatically, but the theory is not at the stage where a useful grammar can be inferred in the majority of applications. Miclet describes typical algorithms for inferring regular and context free grammars from finite sets of strings, which the grammar is required to generate. The applications of grammatical inference that have been investigated include the robotic assembly of mechanical parts, the design of interactive software, and a system for learning structured patterns. The most successful current applications of grammatical inference are in speech recognition. The chapter by Baird is more anecdotal. He gives an example of a successful application in which the grammar employed is regular, rather than one of the context free grammars, but in which a large number, 1360, of productions is employed. The grammar was obtained entirely 'by hand'. Syntactic and structural pattern recognition methods are currently employed most successfully in optical character recognition. Baird suggests that another successful area of application will be robot vision.

Other noteworthy applications of grammatical methods described in detail in the book are the recognition of Chinese characters, the analysis of electrocardiograms, and the automatic generation of descriptions of line drawings.

S. MAYBANK
 London

J. DAWES, M. PICKETT and A. WEARING (editors)
Selecting an Ada Compilation System
 Cambridge University Press, 1991, £22.50
 0-521-40498-3

After agreeing to review this book, I was somewhat surprised to be quoted in the Preface as being a contributor! Although I was certainly involved in some initial discussion about the book, I would be amazed if more than a sentence or two has been contributed by me, so I feel that I can reasonably review the work.

Implementations of the Ada language are unusual in at least two respects: many competing compilers are available (even on mainframes) and compilers often generate code for other systems (especially bare microprocessors). Hence it is not easy to choose an Ada compilation system, since many offerings may well seem appropriate. This book therefore attempts to give user guidance so that an informed choice can be made.

The book is divided into three parts. The first part gives the background to the issues which are relevant to choosing a system. The second part gives a series of questions which can be used to distinguish the products on offer. The third (short) part gives a list of potential sources of useful information.

It seems that as the POSIX standard becomes better established, the portability of Ada systems will grow, and therefore the potential choices for systems to mount applications will increase. The OSF moves towards a standard for programs at the binary level will even make it possible to provide large CASE systems on different platforms, increasing the user choice significantly. Hence books like this one could be a growth area.

An issue at the Ada system level is the support given for tasking on machines with an unconventional architecture. For instance, given a system running on a tightly coupled series of transputers, there is a severe design problem in allocated Ada tasks to specific processors. Dynamic allocation is likely to be too expensive, while static allocation runs the risk of being highly non-optimal and hence giving rise to large data-transfer overheads. I felt that the book did not treat this area well – which unfortunately reflects the current state of Ada compilers for such architectures.

One feature of the Ada compiler market is that virtually all Ada compilers are validated. This does not guarantee correctness, but at least it means one is broadly comparing like with like. It also does not mean that the 'validated' compiler you have purchased will actually pass the ACVC tests, since there is often a slight risk that a correction to the compiler to repair a reported error has actually introduced a further error. The question section therefore asks: 'Given an error, how long would it take before a compilation system is released without it?' I would be worried about too short a period, since it might indicate that the ACVC tests had not been run. Nevertheless, it seems to me that this question is one that can be applied to software more generally.

I can recommend the book to those wishing to understand the issues involved in choosing an Ada compilation system. The book should improve the competitiveness of the market, which should benefit all users.

BRIAN WICHMANN
 Middlesex