ation of Multicomputers for Image Processing, edited L. Uhr *et al.*, pp. 239–278. Academic Press, Orlando, Florida (1986).
10. R. L. Shoemaker, P. H. Bartels, H. Bartels, W. G. Griswold, D. Hillman and R. Männer, Image-data-driven dynamically-reconfigurable multiprocessor system in automated histopathology. In M. J. B. Duff *et al.* Eds.: *Architectures and Algorithms for Digital Image Processing*, edited *Proc. SPIE Conf. 596*, 190–198 (1986).
11. J. Bille, H. Scharfenberg and R. Männer, Biological dosimetry by chromosome aberration scoring with parallel image processing with the Heidelberg POLYP polyprocessor system. *Comput. Biol. Med.* **13** (1), 49–79 (1983).
12. R. Männer, B. Deluigi, W. Saaler, T. Sauer and P. v. Walter, The POLYBUS – a flexible and fault-tolerant multiprocessor interconnection. *Interf. in Comp.* **2** (1), 45–68 (1984).
13. *Motorola: MC68020 32-Bit Microprocessor User's Manual.* Prentice-Hall, Englewood Cliffs, N.J. (1985).
14. *Advanced Micro Devices: The Am2900 Family Data Book* (1979).

Book Reviews

P. B. ANDERSEN
A Theory of Computer Semiotics
Cambridge University Press, 1991, £30.00.
ISBN 0 521 39336 1

From the structuralist and functionalist viewpoint, language is an instrument for action by people in context. Andersen illustrates this with a sample of language used by two mechanics performing an intricate car-repair task. Their linguistic fragments are totally disjointed according to the generative and logical paradigms, but, treated as part of a structure incorporating the users, their materials, tools and task objectives, their language is clearly well formed. Its purpose is to assist in constructing solutions. The relationships between situation and purpose and the making of meaning are more fully illustrated in the major case he uses concerning operations within the Postal Giro in Stockholm.

Among the important concepts for everyone working in the computing field, Andersen introduces the concept of a very specific form of language that we need for systems design. 'A register is the language used in a particular type of situation with the purpose of supporting or changing its activites.' Thus the expressions in a register can be given meanings that are closely related to the actions performed by the users in the given type of situation.

The principal message that emerges in the book is that we should use the register as the definition of users' requirements when designing a wide range of systems. Andersen defines a computer-based register as 'the union of an interface and a work language'. He outlines a method of analysis which focuses upon the design of the signs at the interface, and ensures that they have a 'real meaning' defined by the task to be performed, leaving their 'formal meaning' in computational terms to be implemented by the programmer. This turns the program design inside-out. Instead of thinking that the essence of a program is its functionality, which the interface then reveals, the semiotic designer will regard what happens at the interface as the essential part, to be enabled by the functionality. The challenge to readers convinced by this argument is how to make this new paradigm operational in the normal practice of systems development.

I must add one small criticism. The rule of good interface design should apply to the printed page as well as to the flickering screen. In this case we are shown too often how not to apply them. The typography is a mess.

RONALD STAMPER
Twente, The Netherlands

W. A. HALANG and A. D. STOYENKO
Constructing Predictable Real-Time Systems
Kluwer Academic Publishers, Dordrecht, 1991. £49.75. ISBN 0-7923-9202-7.

This book is most welcome as it covers a number of important topics and issues that are not covered in other books on real-time systems. Primarily the book describes, in detail, two real-time programming languages: Real-Time Euclid and PEARL. The real-time extensions to Euclid include many specific facilities that enable the temporal behaviour of concurrent programs to be controlled. Language primitives are defined that allow, for example: periodic processes to be specified, time bound to be placed on loops, deadline overrun to be detected (and responded to), and sporadic processes to be associated with events. It is unique in supporting schedulability analysis within a language framework. Unfortunately, although Real-Time Euclid is often described as the procedural language which best supports real-time programming it is only a research language. It has not been used in an industrial environment and production-quality compilers are not available.

By comparison PEARL is used in an industry setting. It has a national standard (in fact it has two: PEARL and Basic PEARL) and it is used extensively in process control in Germany. PEARL (Process and Experiment Automation Realtime Language) was designed in the early seventies and includes constructs for time- and/or event-related processing, suspension and termination, and the direct manipulation of hardware. Most previously published descriptions have been written in German, and hence this book will help to publicise PEARL.

In addition to discussing the two languages, other interesting chapters address the issues of scheduling and implementation. A language-independent review of scheduling analysis is given, which attempts to introduce the major results in this field. Such a chapter will always be difficult to write, as it is easy to fall into the trap of providing no new material for the informed reader but too much for the uninitiated. In general the authors manage to strike a reasonable balance. Further chapters on hardware architecture, kernel design and organisation provide much useful, and detailed, information. Topics such as how to use DMA without cycle stealing are covered; this is an issue that has caused considerable difficulty to other architectures. To partition the work between application processing and system support, a dual processor node is advocated. One (simpler) processor undertakes all the scheduling and interrupt handling; this leaves the main processor to be dedicated to application processing. This dual processor model is becoming a common feature of real-time architectures.

One of the themes of the book is the unification of Real-Time Euclid and PEARL to form a language that is well endowed with real-time features but which is standardised and usable for industrial applications. The proposed language, extended PEARL, is described in detail and it is shown how it could support precedence relations, graceful degradation and transient overloads. Even software diversity for tolerating design errors is addressed. It is interesting to compare these proposals with the current debate about Ada9X; a number of similarities exist, for example the introduction of a lock with non-pre-emption for shared data is very close to the protected record with ceiling priority in Ada9X. The extensions to Ada have been criticised for being too ambitious; it will be interesting to see how the proposed changes to PEARL are received.

My main reservation about the book is in its analysis of other languages used in the real-time domain. The authors have been involved with Real-Time Euclid and PEARL for many years, so their evaluation is understandably based on the provision of these languages. When comparing other languages with Real-Time Euclid it is not surprising that Real-Time Euclid wins first prize. They argue, for example that real-time languages must prevent the use of unbounded loops and recursion. But it is equally valid to argue that a more general-purpose language, such as Ada, can be used for real-time applications if it is used in a restricted way in that domain. Although there is no scheduling analysis that can be applied to arbitrary Ada programs, if Ada is used in a suitably constrained way appropriate scheduling theory does exist and can be applied. The book did not need to include this language evaluation; it is valuable in its own right as an information source on two significant languages and their implementations.

ALAN BURNS
York