Abstract
This paper is divided into three sections. The first gives an overview of the presently available and emerging dialects of LISP and how design decisions within them affect symbolic algebra. The second discusses recent developments, in particular, Common LISP, subsetting, portability, pure language research and mixed paradigm systems. The third part is devoted to what is happening in specialised LISP hardware in Japan, in the United States and in Europe. The subject matter of each of these three sections is so tightly interwoven however that the detailed discussion of some material may be postponed until a later section although to some readers it might seem appropriate for inclusion earlier. It should also be mentioned that this is a survey; therefore the items have been selected for mention on the grounds of interest rather than completeness.

I - LISP dialects and algebra support

I.1 Dialects
It has been remarked that whilst in other languages standardisation is regarded as a healthy sign, in LISP it is the reverse; diversity denotes vigour. That observation is patently true today. There are very many varieties of LISP available. too many (and too tedious) to cover in every respect. The systems we do mention are included for some novel aspect of the implementation, significance to the language as a whole or ease of portability and, hence, range of availability.

The systems divide themselves into three distinct groups:

(i) those that have their roots in Project MAC and support, or will be able to support, MACSYMA [Pavelle & Wang, 1985] or the variant of that system known as VAXIMA [Fateman, 1980]. These LISPs include MACLISP [Pitman, 1982]. NIL [White, 1979]. FRANZ LISP [Foderaro & Slower, 1980]. ZETALISP [Weinreb & Moon, 1981] and Common LISP [Steele, 1984].

environments.

(iii) amongst the rest there are several other dialects that should be noted:

UTILISP [Chikayama, 1981] from the University of Tokyo apparently runs some form of MACSYMA, but since UTILISP has no multiple precision integers, it is not clear how useful this is.

LeLISP [Chailloux, 1984] is largely an exercise in porting LISP whose aims are not dissimilar to those already achieved by the Portable Standard LISP project at Utah. LeLISP does not have multiple precision integers and no attempts have been made to put any algebra systems on top of it.

InterLISP [Teitelman, 1978] is probably one of the most well known and longest lived LISP systems. However it has been used, almost exclusively, for artificial intelligence applications. REDUCE was ported to InterLISP-D a few years ago, but the resulting performance was unacceptable.

LISP/VM, [Blair, 1978], which has been known under other names, is one of the older LISP systems in existence, but has only recently become available outside the research establishment at Yorktown Heights (IBM) which developed it. A remarkable feature of LISP/VM, when compared against most other LISP systems, is its semantic completeness and the fact that interpreted and compiled code really are conformant - except in the case of MACROs, which is to be expected. LISP/VM is the base for the Scratchpad algebra system [Jenks, 1984].

In general, the MACLISP family supports MACSYMA and the Standard LISP family supports REDUCE. There is an exception to this categorisation, which is Franz LISP. This now supports both VAXIMA and REDUCE (ported by Toktronix Corporation).

1.2 Power and Efficiency
The above dialects exhibit many variations in the tradeoff between power and efficiency. The major requirement of algebraic computations is high efficiency, because the problems to be solved, although often well understood mathematically, involve a very great amount of work. Expressive power in the language and a complex user environment are nothing without efficiency. Probably the most detailed investigation into this empirical question is that carried out by Richard Gabriel [Gabriel, 1985]. The matter of efficiency must, of course, be affected by dialect considerations such as the long standing debate between speed and purity with respect to the function cell, the capability for environment capture, the conformance of the compiler and the