Ten15 as a basis for PCTE(+) implementation

P. W. Core

R.S.R.E. Malvern

Ten15 - an advanced Portable Systems Kernel

Ten15 is an abstract machine. The term 'abstract' is used to indicate that Ten15 is not associated with any particular machine, rather that it is defined mathematically. The term 'machine' is used to indicate that Ten15 can be thought of as something that can be programmed to manipulate data. As with conventional machines, the nature of the programs that can be accepted by the Ten15 machine and the data types that it can manipulate are extremely important. It is these that determine the programmability of the machine.

Ten15 can be implemented on most modern architectures. There exists almost complete implementations on the RSRE Flex machine and on the DEC Vax machine operating under VMS. The implementation consists of a translator, which translates Ten15 programs into the order code of the target machine, and a run-time kernel which is a set of routines on the target machine which can be called by an executing Ten15 program. The size of the VAX translator is approximately 100 Kbytes and the run-time kernel is less than 200 Kbytes. The estimated effort for writing a translator and kernel is 2 - 3 man years. This is for a maintainable system onto a machine where a model already exists. For example, the VAX translator is a model for say a 68000 translator.

The programs of the Ten15 machine are represented by elements of an algebra. Programs are created by applying the operators of the algebra. On machines these are represented by procedure calls. There already exist Pascal and Algol68 compilers and much of an Ada compiler, compiling programs into Ten15 code. The writing of an ML compiler is planned. A measure of the run-time efficiency of Ten15 can be obtained by comparing the execution of programs compiled directly into the code of the target machine and those compiled via the Ten15 code. Ten15 provides extra facilities such as integrity, mixed language working and fast process management. These place an overhead on every program being executed via Ten15. Executing programs making limited use of these facilities exhibit the worst performance of Ten15. Early tests show that in such cases there is only a 20% increase in the execution time of the program.

The values manipulated by the Ten15 machine include filestore values, processes and communication channels, values accessed over a network and exception values. The inclusion of such and other values allows the expression of an entire operating system as a Ten15 program. Ten15 provides more than this. The existence of Algol68 and Pascal compilers to Ten15 demonstrates that all Algol68 and Pascal programs are expressible as Ten15 programs. Not only does Ten15 provide a basis for systems, but tools built on top of that system. The ability to write a Ten15 translator and kernel for a machine gives the ability to port Ten15 to that machine.

The smallness of that task makes it realistic, especially when compared with the quantity of software that can be ported as a result.

The Ten15 machine is strongly typed, i.e. it uses types to control the application of operators to values. Strong typing is the mechanism used by Ten15 to obtain high levels of efficiency, communication and integrity. Communication and integrity provide a powerful basis for tool creation and composition. This not only has advantages for users of Ten15, who can benefit greatly from tool composition, but it has enabled the design of Ten15 to be in terms of simple well understood operators, knowing that the more complex systems, such as PCTE, can be built by composing tools safely.

Ten15 mechanisms relevant to PCTE(+)

The portability provided by Ten15 has already been discussed, so this section is devoted to two aspects of PCTE(+): its filestore model and its process model.

Filestore

There are two main sorts of filestore object in Ten15. These are the disc variable and the disc pointer. The only sort of object that can be stored in a disc variable is a disc pointer and a disc pointer can point to any type of object. Ten15 has type constructors for these two sorts of values, *pvar* and *persistent*. A value of type *persistent X* is a disc pointer to a value of type *X*, where *X* is any legal Ten15 type. A value of type *pvar X* is a disc variable holding a value of type *persistent X*. The area of disc pointed to by a disc pointer cannot be overwritten; the only way of making changes to the disc is by assigning disc pointers to disc variables. This makes the Ten15 model of filestore essentially non-overwriting. The Ten15 filestore is a persistent heap, with the operations to write and read objects to and from filestore being explicit. Ten15 includes a fast disc garbage collector.

This model of the filestore allows users to write large quantities of data to filestore in the form of disc pointers, but not actually change the filestore until an assignment to the disc variables has been done. If the machine should crash in the middle of altering the filestore, then the filestore would be in either of the state before or after the assignment to the disc variables.

Ten15 allows the updating of several disc variables atomically and it is intended to provide a mechanism for updating disc variables distributed over different filestores atomically. These facilities enable the efficient implementation of distributed transactions.

The ability to write any value to a filestore (done by the writing of efficient coding routines), gives the ability to create complex, possibly fine grained, structures on disc. For example, procedures can be used to enforce schema, disc pointers themselves can be used to represent relationships and disc variables enable the creation of cyclic structures.

Processes

New processes on the Ten15 machine are created by launching a procedure; this procedure is then executed in parallel with and, subject to any communication links, independently of existing processes. It is possible for processes to communicate by common reference, and Ten15 provides queues as an alternative mechanism. When a queue is created, it is specified what type of value can be stored in the queue. This type can be any of the Ten15 value types. Processes can deposit values of the corresponding type onto a queue without waiting and processes can take values off a queue, waiting if there are currently no values in the queue. .

The times to create a process, swap runnable processes and the time to communicate using a queue are approximately the same as the time to execute a procedure call and exit. On the micro VAX implementation of Ten15 this is approximately 50 microseconds. This enables the scheduler to swap processes with a high frequency, a rate of 20 times per second placing an overhead of only 0.1%.

Advanced features supported by Ten15

The Ten15 machine has developed from the examination of languages. It has been extended in a uniform manner to include system programs in addition to those of existing languages. This has enabled system programs to be written making full use of facilities previously only available to the program languages. For example, on a Ten15 machine it is possible to create a system that passes data safely between programs by reference rather than copying. This is the direct analogy of passing reference parameters to procedures in programs, it has been implemented by providing the system with a common address space the same as is available to programs. It is possible that such features could be added to PCTE in the future.

Ten15 is a suitable basis for the writing of tools. It not only provides portability for those tools, but allows the safe communication of complex structured data objects between tools. Any of the Ten15 types can be used as an interface type for tools. These include procedures, references, cycles and polymorphism. The use of typed communication could be incorporated into a system built on top of Ten15.

The high level nature of the data types and the program structure of Ten15 make it a good environment for mixed language programming. The inclusion in the Ten15 machine of its own memory management system together with garbage collector, procedure values and polymorphism provides good support for modern high level languages.

The mathematical form of the Ten15 programs allows tools and programs expressed in Ten15 to be analysed, possibly for the detection of certain kinds of errors. At present there is active research in techniques of program transformation based upon the mathematical structures in Ten15.