FEATURES OF THE GIER ALGOL 4 SYSTEM

PETER NAUR

Abstract.
A brief survey is given of the new features included in the Gier Algol 4 compiler system, as compared with its predecessors. As background the more recent additions to the hardware of the machine are described. The new features primarily aim at a more effective use of the existing machine facilities, including optional facilities existing only in some installations.

Introduction.
Since 1962, when the first version of the Gier Algol system was developed, three new versions, incorporating various improvements over the original design, have been completed. Of these the version known as Gier Algol 4, completed in 1967, represents the greatest step forward from its predecessor. The following paper is an informal description of the new features introduced in Gier Algol 4. A full, user-oriented description is given in the manual of the system [1].

Background.
The new features of Gier Algol 4 must be seen on the background of the characteristics of the machine and the previous Gier Algol systems as reported in ref [2]. The main justification for the new version was the extension of some of the Giers with additional units as follows:

1. A so-called buffer store of 4096 words of core store. Because of the limits of the basic addresses of the Gier, this store has some of the characteristics of a peripheral unit, but it does allow access to single words within 100 microseconds.
2. Disk file units of 384000 words or more.
3. Magnetic tape units.

Another justification was the desire to improve the efficiency of the system for general and administrative data processing. In fact, while the system had proved highly useful for scientific, engineering and experimental data processing uses, lack of effective handling of parts of words had prevented its wide adoption for data processing production programming.
Internal improvements.

Certain improvements of the system do not influence the writing of programs directly, but only improve the service to the users offered by the system. In the translator the most important improvement of this kind is the extension of the check of the match of types to cover actual parameters of procedures called directly (but not of those called via actual-formal correspondence).

Internal improvements of the execution phase include that integers are represented internally by 40 bits, including the sign, and not as floating numbers having only 29 bits in the fixed point part. This is valuable both because larger values can be accommodated, as needed in handling monetary amounts, and because of the higher speed of operations in most cases.

A very important improvement is the use of the buffer store for holding subscripted variables. Previously only up to some 700 variables could be used, and execution speed considerations made it desirable to put the limit even lower, at about 500 variables, cf. [3]. With the buffer store 4096 subscripted variables are allowed, without any ill effects on the speed of execution because of the squeezing of the program.

References to subscripted variables during execution were speeded up in two different ways. First the internal representation of array identifiers was made more convenient for use in the subscription process, by supplying the base address of the array elements and the reference to the storage mapping coefficients in two separate computer words, instead of packing them into one word. Thus the translator already distinguishes two locations for each array. This reduces the time of a reference to a variable having one subscript by about 30 percent. The cost is extra complication in case of array identifiers supplied as formal names, which so far could be treated precisely as non-formal array identifiers. This can be solved in several different ways. As one possibility every reference to such a subscripted variable can be done differently during execution, thus using more time. In order to achieve the highest execution speed we have, however, adopted an alternative solution, to let the procedure call during execution make a complete copy of the description of the array into local locations of the procedure body. The number of locations needed for the description depends on the number of subscripts of the array, however. Within the procedure body this number can only be determined from the uses of the array identifier in subscripted variables. Consider, for example, the following program: