HIERARCHICAL LANGUAGE DEFINITION

Gregor V. Bochmann

Dept Informatique
U. de Montréal

I Introduction:

This paper describes a project for the hierarchical definition and implementation of languages. In the spirit of structured programming [1] and in analogy to the hierarchical construction of operating systems [2,3] we define high level languages, which are suitable for the design of operating and other software systems, in several levels of abstraction. In particular, we describe in this paper how an intermediate, machine independent basic language can be used to express the actions of a given high level language in terms of simple actions of the basic language. In conjunction with a translator writing system, this allows a compact and readable formal definition of the syntax and semantics [4,5] of the high level language, which is in fact a description of the compiler which translates this language into the basic language. This definition is independent of a particular computer, however, an implementation of the basic language must be furnished. We intend to use the basic language to describe modules which are embedded in a general system for multiple processes. The language consists of a kind of macro instructions most of which can be implemented in a straight forward manner. Some more complicated instructions, such as primitives for inter-process communication, can be implemented through a hierarchical construction process [2,3] as indicated in section III (see also figure 1).

II The basic language

Historically, we got interested in a basic language when we tried to express the machine independent semantics of the programming language Pascal [6] in a formal way, using the approach described in references 4 and 7. Our objective is that the basic language

(a) be machine independent, so that a high level language expressed in terms of it can be easily transferred onto another machine.
(b) be simple to understand, and easy to implement on most computers.
(c) be flexible so that it may be used to describe a variety of high level languages, with different types of control structure for sequential processes, including Fortran, Algol, Simula etc.
(d) be a representation which is suitable for all machine independent optimisation, such as the evaluation of constants etc.
(e) contain primitives for inter-process communication (see section III).

Since there is no space to describe the basic language in more detail, we mention the following characteristic points:

(a) Data are represented by bitstrings of variable length. There is also a set of basic specialised data types, such as integer, real, character, etc.
(b) Operations on data, and control instructions are represented by macro-like triplets, such as "addinteger N M".
(c) There are characteristics that remain machine dependent, such as integer precision, character representation etc. They must be described for each particular implementation of the basic language. These machine dependent features also apply for any higher level language which is defined in terms of the basic language (see figure 1).
(d) Variables have an address within a linear address space. Absolute and relative (relative to a base) addressing is possible.
(e) Macros are available for the access of run-time data structures, and for recursive procedure calls. They represent a higher level within the structured programming hierarchy.

III The implementation of multiple processes

The basic language described above allows to express independent parallel computations, and coordinated sequential processes, such as coroutines. In order to implement "really" parallel processes one has to implement an environment for multiple processes, such as described in reference 3, which includes primitives for the execution of parallel processes, and inter-process communication. Hierarchical methods [2,3] for the implementation of parallel processes are known. One could use the following primitives:

(a) control primitives for process execution,
(b) semaphores for shared resourses,
(c) event queues with associated messages for communication among processes, eventually within a network of several computers.

We have explained how the languages used for systems implementation can be constructed in a hierarchical order, such as shown in figure 1. This allows a relatively comprehensive and therefore error-free definition at each level. We try to obtain a system which can be easily transferred onto another computer. In fact, only the compilation of the basic language and the construction of the primitives for parallel processes are machine dependent. Finally, as indicated in figure 1, we propose the implementation of a language for a high level description of parallel processes, as for example given in reference 8.

Acknowledgements: I would like to thank Jack Dennis, Olivier Lecarme, and Jean Vaucher for interesting discussions on this subject.

References:

1. E.W. Dijkstra, Notes on structured programming, in Structured Programming by Dahl, Dijkstra, and Hoare, Acad. Press 1972.

2. W.E. Dijkstra, The structure of THE multiprogramming system, Comm. ACM 11, 5 (May 1968), 341-346.

3. P. Brinch-Hansen, The nucleus of a multiprogramming system, Comm. ACM 13, 4 (April 1970), 238.

4. D.E. Knuth, Semantics of context-free languages, Math. Systems Th., 2, 127 (1968).

5. G.V. Bochmann, Une définition syntaxique et sémantique des langages pour un système d'écriture de compilateurs, Université de Montréal, Document de Travail #30 (1972).

6. N. Wirth, The programming language Pascal, Acta Informatica, 1, 35-63 (1971).

7. G.V. Bochmann, Semantics evaluated from left to right, Publication #135, Département d'Informatique, Univ. de Montréal.

8. P. Brinch-Hansen, Structured multiprogramming, Comm. ACM 15, 7 (July 1972), 574.

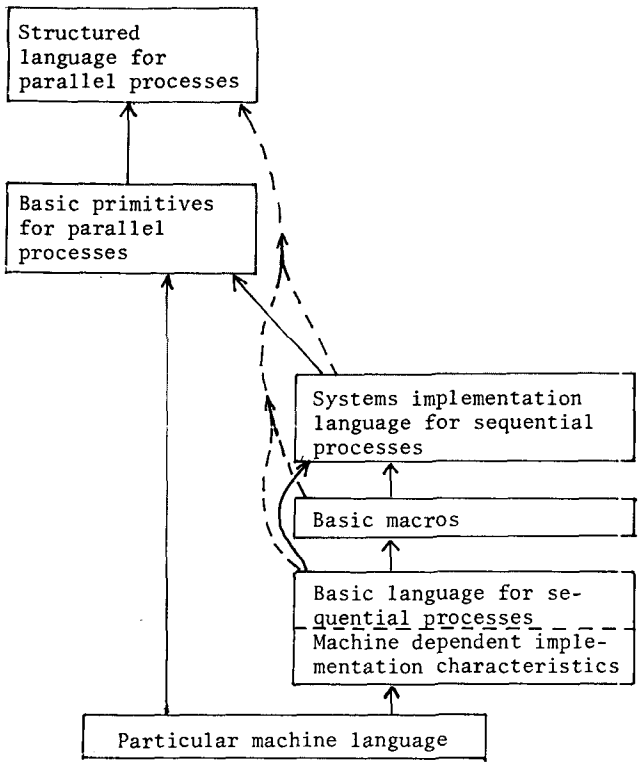

Figure 1: A hierarchy of languages.

Starting from a particular machine language, a number of higher level languages are defined each described in terms of the languages defined previously. Each arrow indicates the usage of a lower level language for the definition of a higher level language. We note that all languages except the particular machine language of the lowest level are essentially machine independent.