

D. Gernert - München

Representation of knowledge by means of nonprocedural languages and their compilers (extended abstract)

Dieter GERNERT, TU München

1. New aspects in computer support to programming

Nonprocedural languages open totally new perspectives in computer support to programming, which can be summed up by the term computer-aided model building. This means that computer assistance is not restricted to the conversion of an existing concept into a computer program, as it is in the usual interactive programming. On the contrary, it is possible to support the model author by confronting him with some knowledge about the underlying section of real world which has been stored in the system (background descriptions).

2. Knowledge represented in a source program

A single source program written in a nonprocedural language may be considered as a problem description, which is built up from three parts: a description of data structures (input and output data), of a situation, and of human valuations (goals, preference structures etc.).

In the general case (where it makes sense to write a program in a nonprocedural language) it is not possible to map a mini-world into mere data structures without a significant loss of information. Therefore, in all nontrivial cases a program written in a nonprocedural language is at the same time a representation of knowledge of the above-mentioned type. By representing knowledge in form of such a program from the beginning, various applications of this knowledge may become easier.

3. Knowledge represented in a compiler

There are several features which distinguish a compiler for a nonprocedural language from a traditional one. These features can be described as different kinds of knowledge represented in the compiler:

1. knowledge about a mini-world (e.g. several kinds of "semantic" errors can be detected automatically),
2. knowledge about problem-solving methods (in the broadest sense, including strategies and algorithms) and about the way of determining the appropriate methods from a given source program (automatic selection of methods),
3. knowledge about relations between problem descriptions (e.g. one problem description can be an "approximation" to an other one, which may be used in the automatic selection of methods).
4. Relations between source programs and hierarchies of descriptions

Among other reasons, the above-mentioned use of background descriptions in computer-aided model building makes it necessary to analyse relations between descriptions. Therefore it is discussed in detail, how a description is composed from "description primitives", which different kinds of relations can exist between two descriptions (especially between two source programs), how a "hierarchy of descriptions" is built up from source programs, and which evaluation processes occur in applying them. Finally it is shown how the concept of a hierarchy of descriptions allows a bottom-up model building in the case of very complex systems.

The complete text will be submitted to the "Bulletin of the EATCS".

Address: Dieter GERNERT,
Schluderstr. 2, D-8000 München 19