TEITELBAUM81

Teitelbaum T., Reps T.; The Cornell Program Synthesizer: A Syntax Directed Programming environment; CACM 24, 9 (September 1981) pp. 563-573.

This paper presents a programming environment which has been developed at Cornell University for use with the programming language PL/CS. The environment is now being used by students at Cornell and a few other universities. The environment aids in the creation, and debugging of programs, both of which can be done interactively. Most of the features used have been included in other systems, but never all at once.

The key to the system is a program editor, which is a cross between a text editor, and a tree editor. The files created by the editor, are not just a stream of text, but information on the structure, content, and method of execution for a program. This file corresponds exactly to a derivation tree of the program, with extra information.

There are two main types of objects when editing, the template, and the phrase. The template, corresponds to a statement, or statement type. The templates cannot be typed in incorrectly, since they are added in total by a single keystroke. The phrases are parts of statements, such as variables, expressions etc. These are typed in by hand. As soon as a phrase has been entered, it is parsed. In addition, it can only be added where it is allowed. The result is, that errors are detected immediately. All editing is done by adding and deleting whole templates, templates cannot be changed, only phrases can be changed.

One very interesting feature is the use of comments to abstract out function. A comment is attached to a set of statements. When viewing the program, the code can be removed, so that only the comment shows, allowing unimportant pieces of code to be temporarily hidden.

The goal was to guarantee that programs are correct. This was eased a bit, so that some errors in phrases can be left in, but the system will flag them. The only errors that can occur are syntactic, or contextual errors in the phrases, since templates are only added where allowed, and are added by the system.

The system updates the file with every change made to the program, and points out errors as soon as they are typed, so the program can be run as soon as it is entered. There is no compilation phase needed, and there is no need for much of the complexity of compilers. (e.g. error correction)

In addition to the program creation tools, there are a number of tools for interactive debugging of the program. During program execution the cursor steps from line to line in the program so that the flow of control can be followed. There is also a facility to display, and update the values of some of the variables used. The programs can be run free, or they can be run at some set speed, single stepped or run in reverse for some small fixed number of steps. This features together allow a very powerful debugging style.

There are a number of small problems with the editor. These are due to the fact that is often often difficult to make a number of seemingly trivial changes. These are caused by the difference between editor's view of the program and that of the programmer.

The system seems to offer some very useful tools, and is definitely an interesting direction, that could well change the view of compiling for many applications.