Setl Newsletter No. 196
Current State of the SETL Implementation

Art Grand
August 28, 1977

We are currently in the process of debugging the new SETL system. At this point we have successfully executed a 200 line program which tests the I/O, all the control structures, arithmetic, and most of the primitives for sets, maps, and tuples. It also tests the garbage collector, including the feature which turns off share bits of objects which are not actually shared. The test program does not contain any declarations since they cannot currently be processed by the compiler.

The run time library is close to completion. The only missing primitives are random, pow and npow. The multiword arithmetic package is still quite crude, and is intended more to handle small negative numbers than to provide general multiword arithmetic. I/O is limited to the SETLA read and print statements; no other I/O has been defined for the language.

At this stage we must put a considerable amount of work into the compiler. Parts of the compiler are over two years old, and have not been adequately maintained as the rest of the system developed. Thus we are unable to parse many language features, such as the new iterater formats and the current syntax for declarations. In addition we have all learned a great deal about programming in the past year. Bugs in the compiler are much more common and harder to fix than those in the library; this suggests that the compiler could well afford some clean up. Finally, we now know the exact tables required by the optimizer. This in itself would require major changes to the compiler.

Most of the work on the compiler will center around writing a completely new semantic pass. Along with this we will write a new grammar as input to the metaparser. This grammar should be much simpler to debug than the current one since it will contain much less backtracking, manipulation of the Polish string, etc. It will also be necessary to do a small amount of work on the parser so that it is consistent with the new implementation of the metaparser.

Finally, it will be necessary to fill in all the missing sections of the code generator which perform special casing. Our approach will be to have the semantic pass perform no peephole optimization whatsoever, but rather to include a pre-pass to the code generator, which may or may not be eliminated once the optimizer is available.

We propose to proceed with this work on the compiler immediately without attempting to run any more test programs or get any timing figures. There are two reasons for this:

1. Most of the bugs we will find by running additional tests will be in the parts of the compiler we plan to rewrite anyway.
2. A number of design decisions in the library were made to favor programs in which all or most of the variables were declared, as opposed to programs in which few if any variables were declared. As long as we can only get timing figures for undeclared programs, we are bound to regret these decisions. This will get us into needless oscilations without complete evidence.