Towards a Static Profiler

Adrian Prantl
Technische Universität Wien

The profiler is an important tool to identify performance bottlenecks in programs. A traditional profiler works by compiling a specially instrumented version of the target program that collects performance data over a series of program executions with typical input data. The collected data is then interpreted and presented to the user as tables and execution graphs.

Since it is up to the user to supply the program with a representative set of input data during the profiling run, it is often left to chance that performance critical paths are triggered during profiling. While this is acceptable for a user interested in the average performance of the program, it is not purposeful for investigating the program's worst-case behaviour; a property especially important to developers of real-time and embedded systems.

In this talk we present the ingredients for a *static profiler* which visualizes an unfolded interprocedural control flow graph by using the results from multiple static analyzers that identify feasible paths and provide upper bounds for loop constructs. The resulting graph is guaranteed to always include the worst-case behaviour of the program.

Our prototype implementation was built using SATIrE program analysis framework as frontend and can analyze C programs. The graph calculation and flow analysis was implemented in Prolog using the Termite library.

References

SATIrE: http://www.complang.tuwien.ac.at/satire/
Termite: http://www.complang.tuwien.ac.at/adrian/termite/