Ideas for Improving Efficiency of Procedural Abstraction

Tamás Szirbucz

Memory consumption in embedded systems is a critical arrea. The smaller code size, the less memory is needed on the device or more features can be implemented in the system. An effective method for reducing code size is the code factoring. An algorithm called procedural abstraction has been previously implemented in GCC [1]. It finds identical copies of code sequences in the program and abstract them out into functions, replacing the original occurences with function calls.

Currently we are investigating how the original algorithm can be made more effective. Below, I present some ideas we are currently focusing on. The original algorithm works on basic blocks. However bigger single-entry/single-exit sections of code can turn out to be clones. Finding such duplicates is computationally more expensive, but can give higher gain as well.

It should also be improved with variable renaming. In many cases two sequences differ in just variable names, that the current algorithm doesn't handle. Otherwise variable renaming increases the overhead of abstractions, and also has higher computing cost.

The current choosing strategy of abstractable sequences is a greedy algorithm. It choose the sequences with higher gain first. I will show, it is less optimal than abstracting the longest sequences first.

At the conference, beside presenting the above ideas, I will discuss implementation problems of these modifications.

References

[1] Gábor Lóki, Ákos Kiss, Judit Jász, and Árpád Beszédes. *Code Factoring in GCC* In Proceedings of the 2004 GCC & GNU Toolchain Developers' Summit (GCC 2004), Ottawa, Ontario, Canada, June 2-4, 2004.