Modeling and Solving Code Generation for Real

Christian Schulte[1,2]

[1] School of ICT, KTH Royal Institute of Technology, Sweden
[2] SICS (Swedish Institute of Computer Science), Sweden
cschulte@kth.se

Abstract. This talk shows how to improve code generation in compilers by using constraint programming (CP) as a method for solving combinatorial optimization problems. It presents how instruction selection (selecting processor instructions for programs), register allocation (assigning program variables to processor registers), and instruction scheduling (reordering processor instructions to increase throughput) can be modeled and solved using CP. The talk covers instruction selection, the integration of register allocation and instruction scheduling, and future plans.

The talk presents a combinatorial model that integrates global register allocation with instruction scheduling. The model covers advanced aspects such as ultimate coalescing, spill code optimization, register packing, and multiple register banks. Unison is a code generator based on the model using CP. Experiments using MediaBench and a processor (Hexagon) that are typical for embedded systems demonstrate that Unison: is robust and scalable; generates faster code than LLVM (up to 41% with a mean improvement of 7%); possibly generates optimal code (for 29% of the experiments); effortlessly supports different optimization criteria (code size on par with LLVM). The model is described in [1, 2].

The talk will sketch a graph-based universal representation that unifies data and control flow for both programs and processor instructions. The representation is the essential prerequisite for a CP model for instruction selection. The model is demonstrated to be expressive in that it supports many processor features that are out of reach of state-of-the-art approaches, such as advanced branching instructions, multiple register banks, and SIMD instructions. The resulting model can be solved for small to medium size input programs and sophisticated processor instructions and is competitive with LLVM in code quality. Representation and model are described in [3].

The models are significant as they address the same aspects as traditional code generation algorithms, yet are based on simple models and can robustly generate optimal code.

The talk is based on joint work with Mats Carlsson, Roberto Castaeda Lozano, Frej Drejhammar, and Gabriel Hjort Blindell.

Acknowledgments. This research has been partially funded by LM Ericsson AB and the Swedish Research Council (VR 621-2011-6229).

References

1. Castaeda Lozano, R., Carlsson, M., Drejhammar, F., Schulte, C.: Constraint-based register allocation and instruction scheduling. In: Milano, M. (ed.) Eighteenth International Conference on Principles and Practice of Constraint Programming. Lecture Notes in Computer Science, vol. 7514, pp. 750–766. Springer-Verlag, Qubec City, Canada (Oct 2012)
2. Castaeda Lozano, R., Carlsson, M., Hjort Blindell, G., Schulte, C.: Combinatorial spill code optimization and ultimate coalescing. In: Kulkarni, P. (ed.) Languages, Compilers, Tools and Theory for Embedded Systems. pp. 23–32. ACM Press, Edinburgh, UK (Jun 2014)
3. Hjort Blindell, G., Castaeda Lozano, R., Carlsson, M., Schulte, C.: Modeling universal instruction selection. In: Pesant, G. (ed.) Twentyfirst International Conference on Principles and Practice of Constraint Programming. Lecture Notes in Computer Science, Springer-Verlag, Cork, Ireland (Sep 2015), to appear.