The Impact of Paradigm Shifts on Safety and Optimality of Code Motion

Jens Knoop and Oliver Rüthing

Universität Dortmund, Baroper Str. 301, D-44221 Dortmund, Germany
{knoop,ruething}@ls5.cs.uni-dortmund.de

1 Motivation

Code motion, or synonymously *partial redundancy elimination* (cf. [3]), is a powerful, classical optimisation, which is widely used in practice. Intuitively, it improves the performance of a program by avoiding unnecessary recomputations of values at runtime. This is achieved by replacing the original computations of a program by temporaries which are initialized at suitable program points as illustrated in Figure 1(a) and (b).

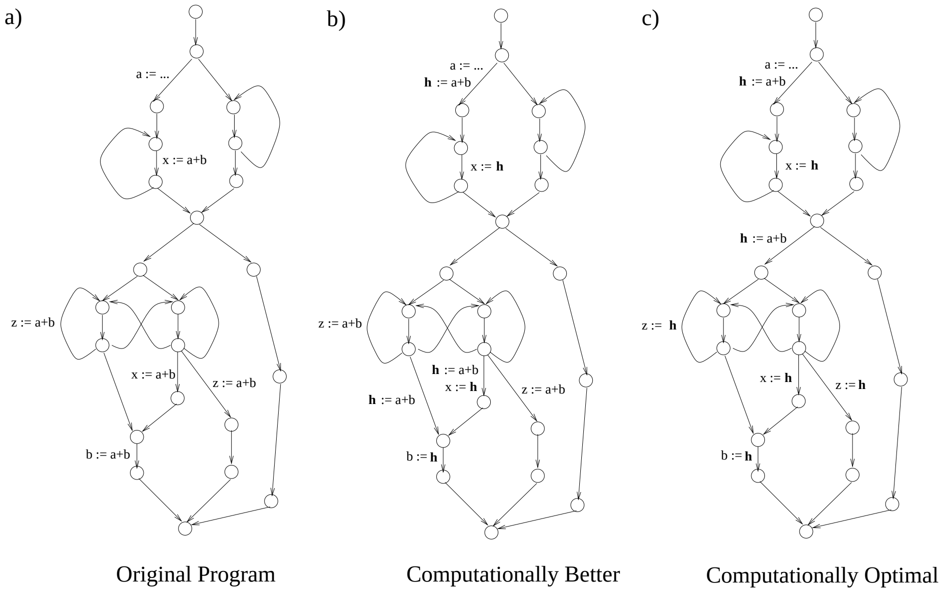

Fig. 1. Code motion: Partial redundancy elimination.

In the intraprocedural setting, it is well-known that even *optimal* results can be achieved. Placing computations as *early as possible* in a program, i.e., without sacrificing its semantics, leads to *computationally optimal* programs (cf. [1, 2]):

on every path from the start node to the end node the program resulting from the as-early-as-possible placement requires at most as many computations as any other program resulting from a *safe*, i.e., semantics preserving, code motion transformation (cf. Figure 1(c)).

In this paper we investigate the validity of the rationale underlying intraprocedurally computationally optimal code motion for various settings and programming paradigms. This ranges from interprocedural programs over explicitly and data-parallel programs to object-oriented programs. In contrast to the intraprocedural setting we demonstrate that the as-early-as-possible placement strategy generally neither guarantees computational optimality nor even safety. Hence, transferring code motion naively from the intraprocedural setting to other settings and paradigms usually fails. However, we show how to adopt the as-early-as-possible placement strategy in order to accommodate for the specifics of the various settings and paradigms under consideration and to make code motion available for these settings, too.

References

1. J. Knoop, O. Rüthing, and B. Steffen. Lazy code motion. In *Proceedings of the ACM SIGPLAN Conference on Programming Language Design and Implementation* (*PLDI'92*) (*San Francisco, California*), volume *27*,7 of *ACM SIGPLAN Notices*, pages 224 – 234, 1992.
2. J. Knoop, O. Rüthing, and B. Steffen. Optimal code motion: Theory and practice. *ACM Transactions on Programming Languages and Systems*, 16(4):1117–1155, 1994.
3. E. Morel and C. Renvoise. Global optimization by suppression of partial redundancies. *Communications of the ACM*, 22(2):96 – 103, 1979.