THE SUMMARY OF Ph. D. DISSERTATION

Major		SURNAME, Firstname
Computer Sciences		Takimoto, Munehiro

Title

Integrated Framework of Code Optimization based on Transforming Dependence Graphs

Abstract

Code optimizations based on code motion have been researched for last decade. For example, partial redundancy elimination (PRE) and partial dead code elimination (PDE) are powerful code optimization techniques based on code motion. PRE not only removes common subexpressions but also eliminates partially redundant expressions, which become redundant on execution of specific program paths. PDE not only removes dead assignments but also eliminate partially dead assignments, which become dead on execution of specific program paths. In addition, they also enable to achieve invariant code motion from loop bodies.

However, code motion techniques have trade-off between precision and efficiency. code motion algorithms using dataflow analysis provide optimal results, which require applying the analysis repeatedly. They are costly. For while, code motion algorithms using dependence graph is more efficient than one using dataflow analysis, because def-use relations are exposed. But they cannot get optimal results.

This paper proposes an efficient and effective algorithm for code motion techniques. Our approach introduce new idea of transforming dependence graph. Since traditional dependence graph only represents dependence structure based on original location of each expression, it cannot provide sufficient precision. Transforming dependence graph corresponding to code motion exposes dependence structures at all program points, which provide def-use information to dataflow

analysis. That enables to achieve the optimal result without repeating application of analysis.

This paper also proposes new dependence graph, Extended Value Graph (EVG) which represents transformed dependence graphs integrally. Since EVG keeps equivalent relation among transformed dependence graphs, their dependence information is combined with dataflow slots.

Transforming dependence graph also extends traditional optimizations effectively. Since this effect can also be captured during building EVG, it can be included into our code motion technique based on EVG.