ABSTRACT

A compiler translates a source code into a target code, often with an objective to reduce the execution time and/or save critical resources. Thus, it relieves the programmer of the effort to write an efficient code and instead, allows focusing only on the functionality and the correctness of the program being developed. However, an error in the design or in the implementation of the compiler may result in software bugs in the target code generated by that compiler. Translation validation is a formal verification approach for compilers whereby, each individual translation is followed by a validation phase which verifies that the target code produced correctly implements the source code. This thesis presents some translation validation techniques for verifying code motion transformations (while underlining the special treatment required on course to handle the idiosyncrasies of array-intensive programs), loop transformations and arithmetic transformations in the presence of recurrences; it additionally relates two competing translation validation techniques namely, bisimulation relation based approach and path based approach.

A symbolic value propagation based equivalence checking technique over the Finite State Machine with Datapath (FSMD) model has been developed to check the validity of code motion transformations; this method is capable of verifying code motions across loops as well which the previously reported path based verification techniques could not.

Bisimulation relation based approach and path based approach provide two alternatives for translation validation; while the former is beneficial for verifying advanced code transformations, such as loop shifting, the latter surpasses in being able to handle non-structure preserving transformations and guaranteeing termination. We have developed methods to derive bisimulation relations from the outputs of the path based equivalence checkers to relate these competing translation validation techniques.

The FSMD model has been extended to handle code motion transformations of array-intensive programs with the array references represented using McCarthy's read and write functions. This improvement has necessitated addition of some grammar rules in the normal form used for representing arithmetic expressions that occur in the datapath; the symbolic value propagation based equivalence checking scheme is also adapted to work with the extended model.

Compiler optimization of array-intensive programs involves extensive application of loop transformations and arithmetic transformations. A major obstacle for translation validation of such programs is posed by recurrences, which essentially lead to cycles in the data-dependence graphs of the programs making dependence analyses and simplifications of the data transformations difficult. A validation scheme is developed for such programs by isolating the cycles in the data-dependence graphs from the acyclic portions and treating them separately. Thus, this work provides a unified equivalence checking framework to handle loop and arithmetic transformations along with recur-

rences.

Keywords: Translation Validation, Equivalence Checking, Bisimulation Relation, Code Motion Transformation, Loop Transformation, Arithmetic Transformation, Recurrence, Finite State Machine with Datapath (FSMD), Array Data Dependence Graph (ADDG)