The Effects of the Precision of Pointer Analysis *

Marc Shapiro and Susan Horwitz
University of Wisconsin – Madison
1210 West Dayton Street, Madison, WI 53706 USA
{mds, horwitz}@cs.wisc.edu

Abstract. In order to analyze programs that manipulate pointers, it is necessary to have safe information about what each pointer might point to. There are many algorithms that can be used to determine this information, with varying degrees of accuracy. However, there has been very little previous work that addresses how much the relative accuracies of different pointer-analysis algorithms affect "transitive" results: the results of a subsequent analysis.

We have carried out a number of experiments with flow-insensitive and context-insensitive pointer analyses to address the following questions:
- How are the transitive effects of pointer analysis affected by the precision of the analysis?
- How good are the "direct" effects of pointer analysis (the sizes of the computed points-to sets) at predicting the transitive effects?
- What are the time trade-offs?

We found that using a more precise pointer analysis does in general lead to more precise transitive results. However, the magnitude of the payoff in precision depends on the particular use of the points-to information. We also found that direct effects are good predictors of transitive effects, and that increased precision in points-to information not only causes a subsequent analysis to produce more precise results, it also causes the subsequent analysis to run faster.

1 Introduction

Compilers often perform a variety of dataflow analyses, such as live variable analysis, to permit safe code optimization. Such analyses can also be used by other programming tools; for example, to provide feedback about possible logical errors, to aid in program understanding and debugging, or to aid in testing. In the presence of pointers, these analyses become more difficult. For example, consider the following code fragment:

```c
x = 0;
printf("%d", *p);
x = 1;
```

If variable p can point to x during the execution of these statements, then x is live after the first assignment, and the assignment itself is considered to be live. If p cannot point to x, then the first assignment is dead (because of the subsequent assignment, x = 1); a compiler can safely ignore the first assignment, and a programming tool might report it as indicating a possible logical error.

* This work was supported in part by the National Science Foundation under grant CCR-9625656, and by the Army Research Office under grant DAAH04-85-1-0482.
To gather the information needed for such analyses, the compiler can first perform a pointer analysis to determine, for each dereference of each pointer $p$, a safe approximation to the set of objects to which $p$ might be pointing at that dereference.

Previous work on pointer analysis has concentrated on the design and analysis of new algorithms. While such work often includes some data produced by running the new algorithm, this data is usually limited to reporting the times required to analyze programs, plus some information about the sizes of the points-to sets that were produced (possibly comparing the times and sizes of the new algorithm to those required/produced by some other algorithm). (For examples, see [LR92], [And94], [Ruf95], [Ste96], [EGH94], [WL95] or [SH97].) While this is certainly interesting, it is not clear whether this information can be used to predict the usefulness of the pointer-analysis algorithm in a larger context. For example, in the case discussed above, we don't really care about the size of $p$'s points-to set, all we care about is whether $p$ might point to $x$.

For this paper we have carried out a number of experiments to address the following questions:

- How are the transitive effects of pointer analysis affected by the precision of the analysis? For example, how much benefit is gained by using a more precise pointer analysis as the first step in solving a dataflow problem such as live variable analysis?
- How good are the "direct" effects of pointer analysis at predicting the "transitive" effects? For example, what is the correlation between the ratio of the sizes of the points-to sets computed by two different pointer analyses, and the sizes of the live variable sets computed using the results of the two pointer analyses?
- What are the time trade-offs? For example, is the extra time required to compute more precise points-to information offset by a decrease in the time required for the subsequent dataflow analysis?

To carry out our experiments, we implemented:

- Four different pointer analyses (described in Section 2.1).
- Three different dataflow analyses (described in Section 2.2), each of which makes use of points-to information.
- Interprocedural slicing using system dependence graphs (described in Section 2.3). Points-to information is used to build the system dependence graphs.

We measured how much the precision of the different pointer analyses affected the results of the dataflow analyses, the sizes of the system dependence graphs, and the results of slicing. In all cases, we also measured how much time was required for both the pointer analyses and for the subsequent processing that made use of the points-to information.

Our results (presented in section 3) can be summarized as follows:

- Using a more precise pointer analysis does in general lead to more precise "transitive" results. However, the magnitude of the payoff in precision depends on the particular use of the points-to information. In our experiments, the difference in precision ranged from 0% to 800%.
- "Direct" effects (points-to set sizes) are good predictors of "transitive" effects (e.g., sizes of live variable sets). However, since the effects of the precision of pointer analysis vary depending on how the points-to sets are used, halving the sizes of the points-to sets does not necessarily lead to a doubling in the accuracy of the transitive results. In our experiments, we found that the expected improvement in indirect effects due to doubling the accuracy of the pointer analysis ranged from 0% to 42%.
- Increased precision in points-to information not only causes a subsequent analysis to produce more precise results, it also causes the subsequent analysis to run faster. (Whether this offsets the longer times needed for the more precise pointer analysis depends, of course, on the relative time requirements of the pointer analysis and the subsequent analysis.)