Milestone Report

Samir Jindel Logan Brooks

{sjindel, lcbrooks}@andrew.cmu.edu

Web page: http://www.andrew.cmu.edu/user/lcbrooks/15745project/

Major Changes

We have made several changes to our original goal. We encountered difficulty creating an analysis in the style of existing points-to analysis that would identify recursive acyclic structures (which are the only interesting such structures). Rather, we determined that we would need a more sophisticated analysis in the style of shape analyses. Consequently, the "Formalize Analysis" task on our roadmap took significantly longer than expected.

Since the analysis we have arrived at is much more complicated than traditional points-to analyses, it will take more time than anticipated to implement, and may require some programmer annotations. In light of this setback, we believe a more realistic goal for our project at this point is either completing the analysis to the extent that it would be useful for a compile-time garbage collector, or implementing a compile-time garbage collector with what we have now, which would only be able to handle a limited set of programs.

While we originally intended to target Java, we became quite frustrated with the existing frameworks for Java analysis, and have decided to analyze the LLVM IR instead. As a result, our analysis may now work on programs compiled from other high-level languages, provided they do not use features that confuse our analysis (such as pointer arithmetic).

What We Have Accomplished So Far

So far we have designed the algorithm to perform the sort of shape analysis required for this application (compile-time garbage collection). We have implemented the data structure used to compress and approximate our view of programs' object graphs at individual points in the programs, as well as the fundamental algorithms on this data structure. The intraprocedural component of the analysis is almost complete.

Meeting Your Milestone

Our milestone was to have completed most of the static analysis; unfortunately we have fallen short of this goal for the reasons in the first section.

Surprises

As mentioned above, the most unexpected surprise we have encountered was that the sorts of algorithms that compute points-to information seemed unable to extract sufficient information to build a useful compile-time garbage collector. We worked around this issue by designed a more powerful analysis that is capable of extracting the necessary information, though it currently requires some programmer annotations. It seems possible to eliminate these annotations by analyzing the program's object graph at runtime, but we may not have time to implement this.

Revised Schedule

Table 1 describes our schedule for the rest of the project.

Week	Samir	Logan
4/16–4/23	Interprocedural Analysis	Finish intraprocedural analysis
4/24–4/30	Implement tools for running the analysis and viewing the output	Debug analysis on test programs

Table 1: Proposed project schedule

Resources Needed

We are not in need of any additional resources at this time.