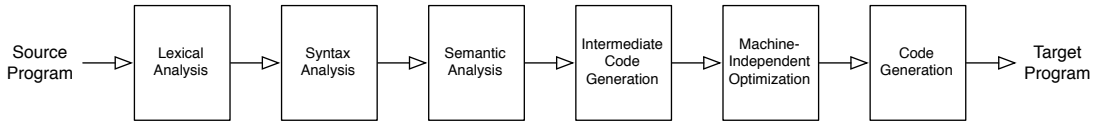
HW 12: The Final HW

CSCI 434T
Fall, 2011

Overview



Rather than assign specific problems to work out this week, I've collected a bunch of papers related to various aspects of compilation. These fill in some gaps, point to current research directions, and will hopefully give a slightly broader perspective on the course. For each topic, I've listed a general, broad overview, as well as one more detailed technical article.

Readings

- Register Allocation
  - Dragon 8.8
- GC
  - "Uniprocessor garbage collection techniques", Paul Wilson.

    A thorough overview of collection techniques. Feel free to skip 3.3–3.6.
  - "Garbage Collection in the Java HotSpot Virtual Machine", Tony Printezis.

    For a more technical discussion:
    `http://www.cs.princeton.edu/picasso/mats/HotspotOverview.pdf`
    or `http://research.sun.com/jtech/pubs/04-g1-paper-ismm.pdf`.
- JITs and Dynamic Optimization
  - Matthew Arnold, Stephen Fink, David Grove, Michael Hind and Peter F. Sweeney. "A Survey of Adaptive Optimization in Virtual Machines". In Proceedings of the IEEE Vol. 93 Iss. 2, February 2005.

    Good for background and an overall perspective on this area. Look for the big ideas, but don't worry about processing all of the details and systems they describe.
  - Andreas Gal et al, "Trace-based just-in-time type specialization for dynamic languages." PLDI 2009.

    Compilation for dynamic languages like JavaScript.

Exercises

1. PA4 Update?
2. Dragon 8.8.1. What is the smallest k such that your graph is k-colorable? If you have $k-1$ registers, what spill code would you add?
3. Be prepared to discuss the general ideas behind the collectors in Wilson.
   - You've probably seen some of these before – mark and sweep, reference counting. What are their limitations?

- What problem does copying-collection solve?
- What is the main insight behind incremental collection?
- What about generational collectors? When will it work well? When will it work poorly?

4. HotSpot JVM Garbage Collection:

   - What assumptions are made about the average Java program? How do these affect the collector design?
   - What is the difference between Serial, Concurrent, and Mostly-Concurrent Collectors? When is each appropriate for a Java program?

5. Dynamic Compilation

   - What's different about optimizing code at run time instead of at compile time?
   - Identify several key areas of run-time optimization for a language like Java.
   - How about for JavaScript? How does the trace-based compilation paper fit into the picture?