Advanced Models and Programs
Some project proposals

Peter Sestoft

2011-04-06

Survey of Java/.NET implementation technology

- (1) Virtual machines and JIT compilers
  - IBM's JVM (Suganuma)
  - Sun's JVM (Lars Bak, Aarhus, et al)
  - Google Chrome's V8 JavaScript VM (Bak)
  - The Mono project's JIT
  - Start with Aycock's survey
- (2) Modern garbage collectors
  - IBM's Metronome for JVM (Bacon)
  - Sun's new garbage collector
  - MS .NET's – if any information is available
  - Start with Wilson's survey

Study Java/C# decompilers

- How good are they?
- How do they work?
- How can one defeat them?
  - Using obfuscators – and how do *they* work?

- Sources (may be outdated):
  - Lutz Roeder's Reflector for .NET
  - Mocha, Jdec, JODE, ... for Java
  - http://www.program-transformation.org/Transform/DeCompilation
  - http://www.backerstreet.com/decompiler/introduction.htm

Compiling MicroC to x86 code

- Idea: Compile MicroC to x86 code
- Topics:
  - Understand the x86 instruction set
  - Decide on runtime (stack) layout
  - Design new Compile methods for MicroC
  - Generate x86 code as text input to nasm
- Challenges:
  - Use extreme care, or spend May debugging
  - Once it works, make it run as fast as possible

Extend MicroC compiler

- Idea: Learn more about C and compilation by covering more of C
- Topics:
  - C features, such as struct types, switch, break, continue, malloc/free, fct pointers
  - Design compile-time type representation and run-time data represenation
  - Modify parser, checker, compiler, (machine)
  - Write example programs and test
- Challenges:
  - Investigate compiler literature and transfer ideas to MicroC compiler

Icon interpreter

- Idea: Icon interpreter in Java or C#
- Topics
  - The Icon language, with backtracking; e.g. (1 to 3) * (4 to 5) gives 4 5 8 10 12 15
  - Continuations – very cool technique
  - Implement parser, optional
  - Implement interpreter using continuations
  - Write and run Icon examples
  - See Programming Language Concepts section 8.9
- Challenges
  - Understand continuations
  - Consult http://www.cs.arizona.edu/icon/
  - Find or invent clever Icon programs
  - (If in .NET, using tail. prefix on calls)

Parallelization of spreadsheet computations

- Starting point: Research prototype in C# of spreadsheet technology (compiled to .NET bytecode)
  http://www.itu.dk/people/sestoft/corecalc/
- Spreadsheets, being functional, offer a lot of almost explicit parallelism

- Idea (1): Target multicore machines, using .NET Task Parallel Library
- Idea (2): Target graphics processors, using Accelerator from MS Research

Specialization of spreadsheet functions, .NET bytecode

- Context: Research prototype of fast sheet-defined functions in spreadsheets
- Implemented in C#
- Challenges:
  - Read up on partial evaluation literature
  - Understand the abstract syntax representation of spreadsheet formulas
  - Understand opportunities for specialization
  - Design and implement specialization, by rewriting/evaluation of abstract syntax

Code Contracts for the C5 library

- Code Contracts: what Joe lectured about, but for C# 4.0
  - invariants
  - preconditions and postconditions
- C5 library:
  - Collection classes and data structures for .NET (Kokholm & Sestoft 2006)
  - 28,000 lines of C# code, but focus on a single data structure (eg. linked lists, trees)
  - Fairly well documented (informally) so there's something to start from
  - http://www.itu.dk/research/c5/

Contact

- Mail sestoft@itu.dk
- Peter's office is 4D15