Lecture 3: The Velocity Compiler

Why a new compiler?

Velocity compiler goals

Velocity is ...

- ... a research compiler
  - human-readable IR
  - convenient for the compiler writer, not necessarily convenient for the user!
- ... a back-end compiler
  - Single, low-level IR
  - Front end provided by other systems
- ... a whole-program compiler
  - supports global analysis, optimization
  - compile-time resource management necessary!
- ... a work in progress
  - many parts not yet available!
- ... a tool for the Liberty group’s compilation research
  - more on the next slide

Why a new compiler?

Velocity compiler goals

- Support for whole-program views
  - compile-time memory management
  - region-based compilation
- No phase ordering artifacts
  - single-level IR
  - incremental analysis
- Machine-independent IR
  - machine descriptions
Velocity Compilation Flow

The Velocity IR: Overview

IR element relationships
- Containment
- Control flow
- Call
- Data flow ...

Let’s look at a real IR file...

The Velocity IR: Implementation

Assignment 1: Dominator Analysis

Step 1: Compute dominator sets
- Dominator sets will be printed in a text file

Step 2: Compute the dominator tree
- Dominator tree will be printed in DOT

Step 3: Identify natural loops
- Natural loops will be printed in different colors, according to their level of nesting

Step 4: Insert loop preheaders
- The new CFG after pre-header insertion will be printed in DOT

Useful classes:
- Liberty.compiler.IR:
  - Module, Procedure, ControlBlock
  - ControlEdge, ControlLink
- Liberty.compiler.View
  - LinearView, CFGView

Implemented in Java
- Java 2 SDK 1.5.0 Early Release used

Let’s look at the documentation ...