Praktikum Optimierende Compiler

Bantam Java Kurzanleitung
Sommersemester 2012

TECHNISCHE
UNIVERSITÄT
DARMSTADT

1 Bantam Java

1.1 Voraussetzungen

1. Plattform: Windows, Linux, Mac
2. Java: JDK 1.7
3. Apache Commons `lang` Bibliothek >= 2.6
   (`http://commons.apache.org/lang/download_lang.cgi`)

1.2 Bestandteile des Bantam Java Pakets

1. `bantamc-lib-obf.jar`
   Zentrale Bibliothek mit dem Kern Bantam Javas (enthält. Lexer/Parser etc.).
2. `spim`
   MIPS Simulator; dient zum Ausführen des von Ihrem Compiler erzeugten Codes.
3. `example driver`
   Beispiel für den Compiler Driver - kann als Vorlage für Ihren Compiler dienen.

1.3 Kurzanleitung: Bauen von `example_driver`

Das grundsätzliche Setup für das Praktikum Optimierende Compiler besteht aus den o.g. Komponenten. Die Kernkomponenten des Compilers (wie Lexer, Parser, AST, CFG etc.) liegen bereits in Form eines High-Level API als Java Bytecode vor und brauchen nicht kompiliert zu werden. Den Compiler Driver — also den eigentlichen Compiler — sollen Sie selbst schreiben, der `example_driver` enthält ein sehr einfaches Skelett, das als Vorlage für die grundsätzliche Organisation des Compilers dienen kann.

1. Um den `example_driver` zu kompilieren, muß lediglich `bantamc-lib-obf.jar` in den classpath aufgenommen werden:

   ```
   javac -classpath <PATH-TO-JAR>/bantamc-lib-obf.jar Main.java
   ```

2. Zum Ausführen muß zusätzlich die Apache Commons lang-Bibliothek im classpath liegen:

   ```
   java -classpath <PATH-TO-JAR>/bantamc-lib-obf.jar:<PATH-TO-LIB>/commons-lang.jar:. Main <BTM-FILE>
   [<BTM-FILE>]*
   ```

3. Bei evtl. auftretenden `ClassDefNotFoundExceptions` zunächst die Pfade prüfen; `HashCodeBuilder` stammt aus der Commons Bibliothek, alle anderen Klassen stammen aus der Bantam Java Bibliothek.

4. Hinweis: Wie bei Java lassen sich Bantam Java Programme auf mehrere Dateien aufteilen, es jedoch nicht zwingend notwendig nur eine (gleichnamige) Klasse pro Datei zu definieren. Angesichts des geringen Umfangs der Programme im Praktikum kann es auch durchaus sinnvoll sein, mehrere Klassen in einer Datei zu definieren (z.B. eine funktionale Klasse und die `Main`-Klasse mit dem Testprogramm). Beachten Sie außerdem, dass die Sprache Bantam Java über keine `import`-Konstrukte verfügt; achten Sie daher beim Aufteilen Ihrer Programme auf mehrere Dateien auf hinreichende Dokumentation, welche anderen Klassen bzw. Dateien benötigt werden / zusammen gehören.

5. Wird der Parameter `-o <OUT-FILE>` nicht angegeben, schreibt `example_driver` den resultierenden MIPS-Assemblercode in die Datei `out.s` im aktuellen Verzeichnis. Die Assemblerdateien können direkt mittels Spim ausgeführt werden (siehe dazu Abschnitt 2).

1.4 Bantam Java API

Das Compiler-Skelett in `example_driver` illustriert bereits die Verwendung der wesentlichen Teile des Bantam Java APIs. Die wichtigste Schnittstelle für die Aufgaben im Praktikum wird das Interface `Optimization` sein; im `example_driver` ist ein prototypische Implementation in Form einer anonymen Klasse enthalten. Die Klasse `Optimizer` unterstützt bei der Durchführung von Passes auf dem CFG: über die Methode `optimize` kann ein CFG-Pass in Form einer `Optimization`-Implementation automatisch auf allen Methoden-CFGs des aktuellen class trees ausgeführt werden.

2 Spim

Die Ihnen bereitgestellte Version von Spim wird mit dem Skript `spim.sh` oder `spim.bat` (Linux/Windows mit Cygwin) gestartet.

Wenn sie als zusätzlichen Parameter den durch Bantam erzeugten MIPS-Assemblercode angeben, wird das Programm direkt gestartet und Sie erhalten im Anschluss eine Übersicht über die Anzahl und Typ der ausgeführten Operationen. Dies können Sie verwenden um die Qualität Ihrer Optimierungen zu überprüfen.