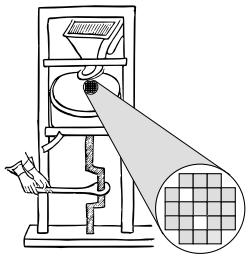
Technische Universität Darmstadt
FG Eingebettete Systeme und
ihre Anwendungen (ESA)

Prof. Dr. Andreas Koch



"Optimierende Compiler"
Aufgabe 4: Optimierung in SSA-Form und Rückwandlung in AST
Abgabe bis zum 12.07.2007, 18:00 Uhr MET DST

1 Einleitung

Als wesentliches Optimierungsverfahren der Vorlesung sollen Sie in dieser Phase die Dominator-basierte Wertnumerierung (DVNT) implementieren. Diese optimiert den nun in SSA-Form vorliegenden CFG. Für die Erzeugung von Maschinencode ist es anschließend erforderlich, den optimierten SSA-CFG wieder in einen AST rückzuwandeln. Dabei müssen in erster Linie die Phi-Funktionen aufgelöst werden.

2 Problemstellung

2.1 Interaktive Oberfläche

Die von Ihnen in Phase 1 erstellte interaktive Oberfläche des Triangle-Compilers soll um zwei Kommandos erweitert werden.

dvnt soll den DVNT-Algorithmus auf die SSA-CFGs aller Prozeduren anwenden. Dabei können Sie mittels eines optionalen Parameters den Namen nur einer zu bearbeitenden Prozedur angeben. Der Name `main` soll dabei für das Hauptprogramm stehen.

ssa2ast soll aus allen SSA-CFGs wieder einen AST erzeugen. Bei Angabe eines optionalen Prozedurnamens als Parameter wird nur diese Prozedur wieder in den AST rückübersetzt, die anderen Prozeduren sollen im AST unverändert bleiben.

2.2 DVNT

Hier soll das in der Vorlesung umrissene Verfahren verwendet werden (im Buch von Cooper & Torczon aus Abschnitt 8.5.2). Dabei muss zunächst aus der IDOM-Relation, die während der SSA-Wandlung nach Brandis und Mössenböck bestimmt werden konnte, ein Dominator-Baum erzeugt werden. Dieser steuert dann die Bearbeitungsreihenfolge (Vorgänger müssen vor Nachfolgern behandelt werden).

Das Paper (auf der Web-Seite) "Value Numbering" von Briggs, Cooper und Simpson kann zum Nacharbeiten der Vorlesung verwendet werden. Auf den Seiten 1 bis 9 beschreibt es die Verfahren allgemein und stellt in Abbildung 4 einen konkreten Algorithmus für DVNT vor.

Dieser weist gegenüber der vereinfachten Variante aus der Vorlesung zwei Erweiterungen auf, die Sie *optional* mit der Möglichkeit der Notenverbesserung realisieren können:

1. Überflüssige Phi-Funktionen können vereinfacht werden. Solche Phi-Funktionen haben dieselbe Wertnummer bei allen Operanden, oder das Ergebnis hat die gleiche Wertnummer wie eine andere Phi-Funktion in diesem Block.
2. Dieser Algorithmus nimmt vor der eigentlichen Wertnumerierung auch noch eine Vereinfachung von Ausdrücken vor (z.B. wird `x+0` zu `x`).

Die erste Erweiterung sollten Sie nach Möglichkeit in Ihrem Code auch berücksichtigen, die zweite ist dagegen weniger wichtig, da sie teilweise bereits von Ihrem `constantfold`-Pass erledigt wird.

Zur Vereinfachung kann sich Ihre DVNT-Realisierung auf die Bearbeitung *vollständiger* Ausdrücke beschränken. Das heisst, dass bei den Anweisungen

```
x := a+b+c;
y := a+b;
z := a+b+c;
```

keine Wiederverwendung eines Teilausdrucks von `x` nach `y`, sondern nur die des vollständigen Ausdrucks von `x` nach `z` erkannt werden muss. Sie können aber *optional*, ebenfalls mit der Möglichkeit der Notenverbesserung, auch die Bearbeitung von Teilausdrücken realisieren.

Neben dem transformierten SSA-CFG, den man sich ja mit `dumpcfg` anzeigen lassen kann, soll Ihre Implementierung ausgeben, wieviele Berechnungen von Ausdrücken vermieden werden konnten (im Beispiel oben also 1) und (falls implementiert) wieviele Phi-Funktionen eliminiert werden konnten.

2.3 SSA-CFG nach AST rückwandeln

Hier soll das im 7. Block der Vorlesung beschriebene Verfahren zur Rückwandlung eingesetzt werden. Da Sie ja (wie dringend empfohlen) schon bei der Konzeption Ihrer CFG-Datenstruktur die Rückrichtung im Auge behalten haben, liegt die Hauptschwierigkeit hier im Auflösen der Phi-Funktionen in geeignet platzierte Kopieranweisungen.

Wie in der Vorlesung beschrieben, müssen dabei kritische Kanten, die ja auch in Triangle auftreten können, durch Aufspalten korrekt behandelt werden.

Die Funktionsweise dieser Phase sollten Sie durch `check`, `codegen` sowie dem Laufenlassen Ihrer Testprogramme überprüfen.

3 Abgabe

Es gelten auch hier die auf dem ersten Aufgabenblatt beschriebenen Anforderungen an **Programmierstil** und **Dokumentation**. Bitte lesen Sie die entsprechenden Abschnitte falls nötig nocheinmal!

Jede Gruppe schickt spätestens zum Abgabezeitpunkt in einem `.jar`-Archiv alle Dateien ihrer Version des Triangle-Compilers an

`oc07@esa.informatik.tu-darmstadt.de`

mit dem Subject `Abgabe 4 Gruppe` *N*, wobei Ihnen *N* bereits in der Vorlesung mitgeteilt wurde. In dem Archiv sollen nicht nur die eigenen, sondern *alle* (auch unmodifizierten) Quellen des Triangle-Compilers enthalten sein. Ebenso legen Sie eventuell verwendete zusätzliche externe Bibliotheken in Form ihrer jeweiligen `.jar`-Dateien bei (aber siehe Abschnitt 6).

In dieser Endabgabe ist nun auch die **Kommentierung** relevant: Jede der von Ihnen modifizierten oder neu erstellten Quelldateien trägt oben einen Kommentarkopf, der die Funktion der Datei sowie die im Laufe ihrer Entstehungsgeschichte vorgenommenen Änderungen dokumentiert. Zu jeder Änderung **muß** der entsprechende Autor angegeben werden.

Neben dem Kopfkommentar versehen Sie auch die einzelnen von Ihnen neu eingeführten oder veränderten Methoden und Instanzvariablen mit aussagekräftigen Kommentaren entsprechend den JavaDoc-Konventionen.

Innerhalb der Methoden beschreiben Sie durch aufschlußreiche Kommentare den allgemeinen Ablauf, der auf einer höheren Abstraktionsebene als die der einzelnen Java-Anweisungen beschrieben werden soll.

Bei der Programmierung verwenden Sie einen einheitlichen, gut lesbaren Stil. Als Vorschlag dazu seien hier die AmbySoft Java Coding Guidelines genannt (auf dem OC07 Web-Site verfügbar).

Neben den Java-Quelltexten enthält das Abgabearchiv eine Datei README.txt, die enthält

- die Namen der Gruppenmitglieder.
- eine Übersicht über die neuen und geänderten Dateien mit jeweils einer kurzen (eine Zeile reicht) Beschreibung ihrer Funktion.
- Hinweise zur Compilierung der Quellen. Geben Sie eine `javac`-Kommandozeile an bzw. verweisen Sie auf mitgelieferte Makefiles oder ANT Build-Dateien. *Nicht* ausreichend ist ein Hinweis auf eine von Ihnen verwendete IDE (wie Eclipse, NetBeans etc.).
- Angaben über weitere Bibliotheken (beispielsweise JSAP, log4j, JUnit etc.), die Sie eventuell verwendet haben. Diese Bibliotheken legen Sie bitte dann auch als .jar Dateien in das abgegebene Archiv.
- Beschreibungen zu etwaigen Fehlerkorrekturen zu den früheren Phasen.
- für alle Beispielprogramme (in `triangle-examples.tar.gz`) die Ausgaben von `dumpcfg` für alle Prozeduren (als einzelne Dateien) nach `read` / `check` / `ast2ssa` / `dvnt`
- für alle Beispielprogramme daran anschliessend die Ausgaben von `showast` und `dumpast` nach `ssa2ast` als einzelne Dateien
- Eine Statistik, wieviele Operationen durch DVNT aus den verschiedenen Beispielprogrammen entfernt werden konnten.

4 Beurteilung

Die Kolloquien zu dieser Abgabe finden am 17.7.07 in der regulären Vorlesungszeit in Raum E103 statt. Diese Kolloquien sind Bestandteil der Prüfungsleistung, es besteht daher **Anwesenheitspflicht** für alle Gruppenmitlglieder.

Ein Kolloquium dauert jeweils ca. 15 Minuten. Anfangszeiten und Gruppennummern sind unverändert vom 2. Aufgabenblatt.

5 Anregungen zur Gruppenarbeit

Bei dieser Phase ist die DVNT-Implementierung deutlich komplizierter als die SSA-CFG-nach-AST-Transformation (zumindest, wenn Sie in der letzten Phase die CFG-Datenstruktur geeignet gewählt haben).

Es bietet sich daher an, ein Mitglied an der Rücktransformation und die beiden anderen an DVNT arbeiten zu lassen. Natürlich ist auch eine klassische Dreiteilung nach dem Schema

- Rücktransformation
- DVNT
- Test beider Systeme

möglich.

Der Tester muss dabei mit der Arbeitsweise beider Verfahren (aber nicht zwangsläufig ihrer Implementierung) vertraut sein und geeignete Testfälle konstruieren, die potentielle Fehlermöglichkeiten weitgehend abdecken. Die Testfälle bestehen jeweils aus dem Triangle-Quellcode, den erwarteten SSA-CFGs vor und nach der DVNT-Optimierung, sowie dem erwarteten AST nach der Rückwandlung. Damit ist dann eine Kontrolle des ganzen Compile-Flusses möglich. Hinweis: Die Beispielprogramme sind als Eingaben für die ersten Tests bereits **viel zu kompliziert**! Schreiben Sie *gezielt* kleinere Testprogramme, die "Futter" für die Optimierung enthalten und auch Sonderfälle (z.B. kritische Kanten) abdecken.

6 Plagiarismus

Der Fachbereich Informatik misst der Einhaltung der Grundregeln der wissenschaftlichen Ethik großen Wert bei. Zu diesen gehört auch die strikte Verfolgung von Plagiarismus. Mit der Abgabe einer Lösung zu den Programmierprojekten bestätigen Sie, dass Ihre Gruppe die alleinigen Autoren des neuen Materials bzw. der Änderungen des zur Verfügung gestellten Codes sind. Im Rahmen dieser Veranstaltung dürfen Sie den Code des Triangle-Compilers vom OC07 Web-Site sowie Code-Bibliotheken für nebensächliche Programmfunktionen frei verwenden. Mit anderen Gruppen dürfen Sie sich gerne über grundlegende Fragen zur Aufgabenstellung austauschen. Detaillierte Lösungsideen dürfen dagegen *nicht vor Abgabe*, Artefakte wie Programm-Code oder Dokumentationsteile *überhaupt nicht* ausgetauscht werden. Bei Unklarheiten zu diesem Thema (z.B. der Verwendung weiterer Software-Tools oder Bibliotheken) sprechen Sie bitte Ihren Betreuer gezielt an.