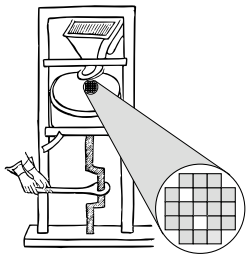
Technische Universität Darmstadt
FG Eingebettete Systeme und
ihre Anwendungen (ESA)

Prof. Dr. Andreas Koch



"Optimierende Compiler"
Aufgabe 4: Optimierung in SSA-Form und Rückwandlung in AST
Abgabe bis zum 12.7.2006, 12:00 MET mittags

1 Einleitung

Als wesentliches Optimierungsverfahren dieser Veranstaltung 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 2 erstellte interaktive Oberfläche des Triangle-Compilers soll um zwei Kommandos erweitert werden.

dvnt soll den DVNT-Algorithus auf den SSA-CFG anwenden

ssa2ast soll aus dem SSA-CFG wieder einen AST erzeugen

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 Verbesserungen auf:

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.

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

Jede Dreiergruppe schickt spätestens zum Abgabezeitpunkt in einem Archiv alle Dateien ihrer Version des Triangle-Compilers an

`oc06@esa.informatik.tu-darmstadt.de`

mit dem Subject `Abgabe 4 Gruppe` *N*, wobei Ihnen *N* bereits via Email mitgeteilt wurde.

In dem Archiv sollen nicht nur die eigenen, sondern *alle* (auch unmodifizierten) Quellen des Triangle-Compilers enthalten sein. Dazu kommen noch die neue(n) Testeingabedatei(en) im Triangle-Subset.

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. Aufgrund dieser Daten erfolgen dann gezielte Nachfragen in den Kolloquien.

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 OC06 Web-Site verfügbar).

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

- die Namen der Gruppenmitglieder.
- eine kurze Beschreibung, was welches Gruppenmitglied zur Lösung beigetragen hat.
- eine Übersicht über die neuen und geänderten Dateien mit jeweils einer kurzen (eine Zeile reicht) Beschreibung ihrer Funktion.
- Hinweise zur Compilierung der Quellen. Hier geben Sie bitte auch an, falls Sie weitere Bibliotheken (beispielsweise JSAP, log4j, JUnit etc.) verwendet haben. Diese Bibliotheken legen Sie bitte dann auch als .jar Dateien in das abgegebene Archiv.

4 Beurteilung

Die Beurteilung dieser Abgabe erfolgt in Kolloquien. Diese finden dann an einem der Dienstagstermine statt. Vorträge sind erst zum Ende des Semesters geplant (nach Abschluß aller Phasen).

5 Anregungen zur Gruppenarbeit

Bei dieser Phase ist die DVNT-Implementierung deutlich komplizierter als die SSA-CFG-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 Quellcode im Triangle-Subset, 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.

6 Ausblick auf Aufgabe 5

In der fünften und damit letzten Phase der diesjährigen Veranstaltung sollen noch Aufräumungsarbeiten im CFG vorgenommen werden. Bei der SSA-Konversion werden in vielen Fällen überflüssige Kopien der Form `x := y` angelegt, die häufig eliminiert werden können. Statdessen werden alle folgenden Benutzungen von `x` durch `y` ersetzt. Das konkrete Verfahren heisst Copy Propagation, ist vergleichsweise einfach, und wird in einer der folgenden Vorlesungen vorgestellt werden.

Ebenso kann es sinnvoll sein, aus dem CFG überflüssige (z.B. leere Blöcke) korrekt zu entfernen. Auch die dafür nötigen Verfahren, namens CLEAN und DEAD (Cooper & Torczon, Abschnitt 10.3.1), sind recht überschaubar und werden noch präsentiert.

Da für diese letzte Phase nur wenig Zeit ist (Abgabe spätestens am 21.7.) bietet es sich an (auch in Anbetracht der Überschaubarkeit der Programmierung) dass nun *alle* Gruppenmitglieder parallel je einen dieser Passes implementieren. Da die Passes unabhängig voneinander arbeiten, ist auch ein getrenntes Testen möglich.

7 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 OC06 Web-Site sowie Code-Bibliotheken für nebensächliche Programmfunktionen (Beispiele siehe oben) frei verwenden. Mit anderen Gruppen dürfen Sie sich über grundlegenden 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 an.