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

Prof. Dr. Andreas Koch

embedded
systems +
applications

"Optimierende Compiler"
Aufgabe 3: Rückwandlung von SSA nach AST
Abgabe bis zum 07.06.2010, 23:59 Uhr MET DST

1 Einleitung

Nun sollen Sie zwei Techniken zu einer Gesamtlösung kombinieren. Es handelt sich dabei um die Rückwandlung aus der SSA-Form nach dem Verfahren von Briggs, Cooper, Harvey und Simpson, aufbauend auf der Datenflussanalyse.

2 Problemstellung

2.1 Interaktive Oberfläche

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

flowlive soll auf die SSA-CFGs aller Prozeduren (einschliesslich des Hauptprogramms) eine Datenflußanalyse ausführen und für jeden Block des CFG die Mengen UEVAR, VARKILL und LIVEOUT berechnen und an die CFG-Knoten annotieren.

dumplive Soll alle Ergebnisse von `flowlive` ausgeben. Bei Angabe eines optionalen Prozedurnamens als Parameter werden nur die Daten dieser Prozedur analysiert. Durch einen optionalen davorstehenden Parameter `-o` *filename* soll die Ausgabe in die angegebene Datei erfolgen.

ssa2ast soll die SSA-CFGs aller Prozeduren (einschliesslich des Hauptprogramms) wieder in die normale Form (ohne Phi-Funktionen) umwandeln und das Ergebnis als für die Code-Erzeugung geeigneten DAST ablegen. Bei Angabe eines optionalen Prozedurnamens als Parameter wird nur diese Prozedur wieder in den DAST rückübersetzt, die anderen Prozeduren sollen im DAST unverändert bleiben.

2.2 Datenflussanalyse

Das `flowlive` Kommando soll seine Ergebnisse in Ihren internen Datenstrukturen ablegen (damit sie weiter verwendet werden könen, siehe unten), aber ohne Angabe einer von Ihnen definierten Debug-Flag *keine* Ausgaben machen (das ist für unsere automatischen Tests wichtig). Die Ausgaben der Ergebnisse soll getrennt durch das `dumplive`-Kommando erfolgen. Dabei soll als Ausgabeformat das im 5. Vorlesungsblock für die Copy Propagation gezeigte benutzt werden. Für einen CFG für die Prozedur foo mit den Blöcken B1, B2 und B3 könnte nach `flowlive` der Aufruf von `dumplive` folgende Ausgabe liefern (sinnlose Beispiele):

```
CFG: foo
UEVar(B1) = {a_2}
UEVar(B2) = {b_3,c_1}
UEVar(B3) = {c_1}
VarKill(B1) = {}
VarKill(B2) = {a_2}
VarKill(B3) = {b_3}
LiveOut(B1) = {}
LiveOut(B2) = {c_1}
LiveOut(B3) = {}
```

Die Versionsnummern der SSA-Wertinstanzen sind durch einen Unterstrich vom Variablennamen getrennt.

Behandeln Sie zusammengesetzte (`record`) und Array-Typen wie in Aufgabe 2 (Erzeugung der SSA-Form) beschrieben. Definieren Sie neben den oben gezeigten interaktiven Kommandos auch ein API, so dass die nachfolgenden Teilaufgaben eine Analyse auslösen können und die Ergebnisse abrufen. Dazu ist es sinnvoll, intern keine Textdarstellung zu verwenden (Strings sind böse!), sondern beispielweise den DAST bzw. Ihren CFG um entsprechend verkettete Objekte zu erweitern.

Zur Lösung der Datenflussprobleme können Sie einen einfachen iterativen Algorithmus verwenden. Eine Optimierung auf niedrige Iterationszahl (durch geeignete Rechenreihenfolge) ist *nicht* verlangt, kann aber optional realisiert werden und könnte damit die Gesamtbewertung verbessern.

Als weitere optionale Aufgabe können Sie `dumpcfg` so erweitern, dass auch die Ergebnisse der Datenflußanalysen an die ausgegebenen CFGs annotiert werden und so auch in den dot-Grafiken auftaucht.

2.3 SSA-CFG nach AST rückwandeln

Hier soll das im 6. Block der Vorlesung beschriebene Verfahren zur Rückwandlung eingesetzt werden. Da Sie ja hoffentlich (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. Verwenden Sie dazu den in der Vorlesung vorgestellten erweiterten Briggs-Algorithmus, der auch die Variablen in Kontrollbedingungen korrekt behandelt. Bauen Sie für die dafür erforderliche Liveness-Analyse auf den Ergebnissen des `flowlive`-Passes auf, den Sie nötigenfalls automatisch aufrufen.

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

3 Hinweise zum Kürzen

Falls Sie sich ausserstande sehen, die Aufgabenstellung im geforderten Umfang zu bearbeiten (aus Zeitmangel, Verständnisprobleme, etc.), sind hier einige Vorschläge für *sinnvolles* Kürzen. Das Implementieren des reduzierten Leistungsumfangs wird aber eine Abwertung nach sich ziehen (z.B. Basisnote nicht mehr 2,0 sondern 3,0 o.ä.).

1. Wandeln Sie den SSA-CFG nicht mit dem Briggs-Algorithmus zurück, sondern mit dem Verfahren aus dem 4. Block der Vorlesung. Vermeiden Sie Fehlerfälle durch das Aufspalten aller kritischen Kanten.
2. Wenn Sie diese Varianten wählen, können Sie auch die Datenflußanalyse komplett weglassen, Ihren Compiler aber wenigstens mit eingeschränktem Funktionsumfang vorführen.

Die angemeldeten Zweiergruppen können als Arbeitserleichterung **ohne Abwertung** die Variante 1 verwenden. Sollte hier das volle Briggs-Verfahren erfolgreich implementiert werden, würde dies zu einer Aufwertung führen.

4 Testen

Um Ihnen das Testen zu erleichtern und das bekannte Prinzip auszunutzen, dass man selbstgeschriebenen Code am besten mit fremdgeschriebenen Testfällen erprobt, dürfen Sie ab jetzt Ihre Triangle-Testprogramme auch gruppenübergreifend **austauschen** (z.B. über das Forum). Sie sollten dabei neben den Programmen selbst auch Angaben über die Ihrer Meinung nach richtigen Ergebnisse machen.

Vergessen Sie nicht, dass Ihr Compiler nach `ssa2ast` nun auch mit `codegen` gültigen Code für Ihren rückgewandelten CFG erzeugen muss. Wenn `codegen` abstürzt oder der optimierte Code (den Sie jetzt ja ausführen können!) falsche Berechnungen durchführt, haben Sie noch Fehler in Ihrem Compiler!

5 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` oder `.zip`-Archiv alle Dateien ihrer Version des Triangle-Compilers an

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

mit dem Subject `Abgabe 3 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 8).

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 (aus dem Verzeichnis `Examples/` der Triangle-Quellen) die Ausgaben von `dumplive` für alle Prozeduren (als einzelne Dateien) nach `read` / `check` / `ast2ssa` / `flowlive`.
- für alle Beispielprogramme daran anschliessend die Ausgaben von `showast` und `dumpast` nach `ssa2ast` als einzelne Dateien

6 Beurteilung

Die Kolloquien zu dieser Abgabe sollen in KW 24 (14.06.-18.06.) stattfinden. Vereinbaren Sie dazu (wie schon für Aufgabe 2) einen entsprechenden Termin im Forum. Diese Kolloquien sind Bestandteil der Prüfungsleistung, es besteht daher **Anwesenheitspflicht** für alle Gruppenmitlglieder.

Ein Kolloquium dauert jeweils ca. 20 Minuten. Anfangszeiten und Gruppennummern sind unverändert von früheren Aufgaben.

7 Anregungen zur Gruppenarbeit

Bei dieser Aufgabe sind *alle* Gruppenmitglieder bei der Implementierung gefordert! Es bietet sich folgende Aufteilung an:

- Datenflussanalyse (`flowlive`, `dumplive`)
- SSA-Rückwandlung (`ssa2ast`)
- Entwicklung von Testfällen für Datenflussanalyse und Rückwandlung, Durchführung der Tests

Vom Programmierumfang sind die Aufgaben ungefähr vergleichbar, `ssa2ast` hat aber den aufwendigsten Algorithmus. `ssa2ast` braucht eine wohldefinierte Schnittstelle zum Datenflussanalysetei. Hier lohnen sich *sorgfältige* Absprachen zu einem geeigneten API! Danach kann auch die Datenflussanalyse unabhängig von den `ssa2ast` entwickelt werden.

Zum Testen verwenden Sie eigene Testprogramme oder die von anderen Gruppen zur Verfügung gestellten. Die Testfälle könnten jeweils bestehen aus dem Triangle-Quellcode, den erwarteten SSA-CFGs 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 auch Sonderfälle (z.B. kritische Kanten) abdecken. Lassen Sie sich für's Erste von den Code-Beispielen des 6. Vorlesungsblockes inspirieren!

8 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 OC 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. Von letzterer Einschränkung explizit ausgenommen sind Triangle-Beispielprogramme zum Testen des Compilers. Diese **dürfen** ausgetauscht werden (z.B. im Forum der Veranstaltung). Bei Unklarheiten zu diesem Thema (z.B. der Verwendung weiterer Software-Tools oder Bibliotheken) sprechen Sie bitte Ihren Betreuer gezielt an.