

Praktikum Compilerbau

Sitzung 7 – Optimierung: Konstantenfaltung

Prof. Dr.-Ing. Gregor Snelting
Matthias Braun und Sebastian Buchwald

IPD Snelting, Lehrstuhl für Programmierparadigmen



1. Letzte Woche
2. Motivation
3. Datenflussanalyse
4. Konstantenfaltung mit libFirm
5. Sonstiges

Letzte Woche

- Was waren die Probleme?
- Hat soweit alles geklappt?

Zwischenstand Gruppe X

Parser, Pretty-Printer

- NullPointerException bei `class {}`
- `class EOF {}` wird abgelehnt!
- Einfache Programme funktionieren, kompliziertere meist nicht:
- `EmptyStatement created in line XXX` Meldungen
 - falsche Klammerung fehlende Strichpunkt
 - zum Teil anscheinend in Endlosschleife (oder extrem langsam)
 - Exceptions bei `NewArrayExpressions`

Semantik

- Fast alles wird abgelehnt wegen angeblich fehlendem `return`.
- Deshalb konnte nicht viel mehr getestet werden.

Zwischenstand Gruppe Y

Parser, Pretty-Printer

- Keine Parserfehler; Alle fehlerhaften Eingaben erkannt
- Pretty-Printer:
 - Argument der main-Funktion immer args
 - Zusätzliche Klammer bei Initialisierung lokaler Variablen
- (Zusätzliche Leerzeile am Ende, RuntimeException wenn Datei nicht geöffnet werden kann, Exceptions mit Backtraces als Fehlermeldungen, kein error-String bei Fehlern)

Semantik

- NullPointerException bei return;
- Scoping: `int x; { int x; }, foo(int x) { int x; }` in Java nicht erlaubt.
- Zugriff auf Felder in main nicht abgelehnt
- (Keine Ausgabe von error bei Syntaxfehlern)

Zwischenstand Gruppe Z

Parser, Pretty-Printer

- -2147483648 wird abgelehnt
- Alle fehlerhaften Eingaben erkannt.
- Fehlende Klammern bei Ausgabe von Call, Assignment
- (Zusätzliche Leerzeile + Leerzeichen)

Semantik

- `foo(null)` abgelehnt
- Scoping: `int x; { int x; }, foo(int x) { int x; }` in Java nicht erlaubt.
- `void`-Typ für Felder, Arrays und Parameter nicht abgelehnt
- `void foo() {} void bar() { return foo(); }` nicht abgelehnt
- `return;` akzeptiert wenn Wert benötigt
- Fehlendes `return` wird nicht immer bemängelt
- (Error statt error)

1. Letzte Woche

2. Motivation

3. Datenflussanalyse

4. Konstantenfaltung mit libFirm

5. Sonstiges

Wozu Optimierungen?

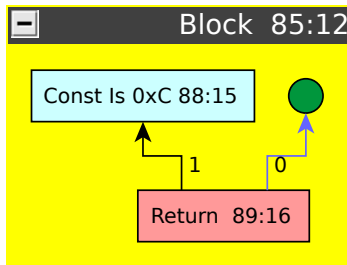
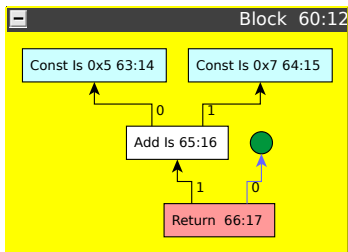
- Programme sollen besser werden
- Mögliche Optimierungsziele:
 - Laufzeit
 - Codegröße
 - Energieverbrauch

Anforderungen an Optimierungen

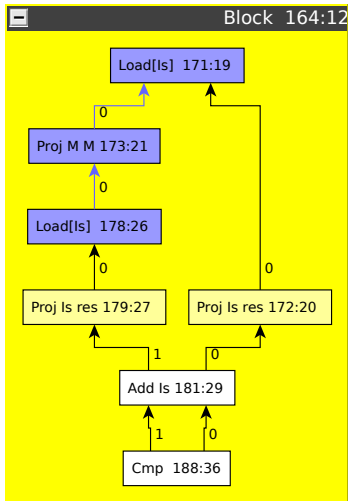
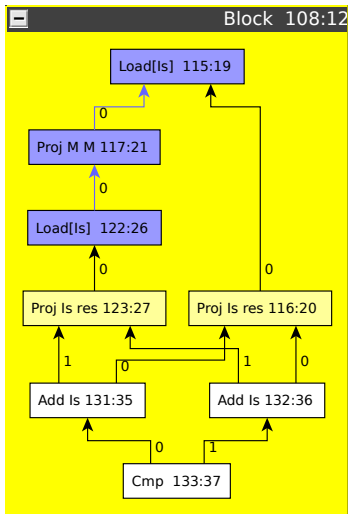
- Korrektheit: Optimierungen sollen Semantik erhalten
- Effizienz: Optimierung muss auch große Programme in angemessener Zeit verarbeiten können

- Konstantenfaltung
- Elimination gemeinsamer Teilausdrücke (engl. common subexpression elimination, CSE)
- ... (optional)

Konstantenfaltung



Elimination gemeinsamer Teilausdrücke



1. Letzte Woche
2. Motivation
- 3. Datenflussanalyse**
4. Konstantenfaltung mit libFirm
5. Sonstiges

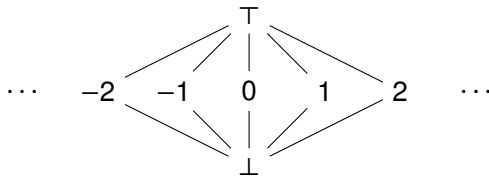
Wozu Datenflussanalyse?

Datenflussanalyse ermöglicht es den Rückgabewert des folgenden Programms zu bestimmen:

```
int optimizeMe() {  
    int x = 1;  
  
    while (...)  
        x = 2 - x;  
  
    return x;  
}
```

- Verband $\mathcal{L} = (L; \leq, \sqcap, \sqcup)$ endlicher Höhe
 - \top ist größtes Element des Verbandes
 - \perp ist kleinstes Element des Verbandes
- Menge von monotonen Transferfunktionen $f : L \rightarrow L$
- Theorie: Es gibt ein i so dass $f^i(\perp) = f^{i+1}(\perp)$

- Verband besteht aus n -Tupeln über dem folgenden Verband



- Punktweise Anwendung von \sqcup bei mehreren Vorgängern

- Transferfunktionen

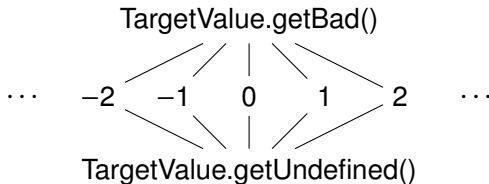
- $f_{x_2=\phi(x_0, x_1)}(l) = l[l_{x_2} \leftarrow l_{x_0} \sqcup l_{x_1}]$

- $f_{z=x+y}(l) = \begin{cases} l[l_z \leftarrow \perp] & l_x = \perp \vee l_y = \perp \\ l[l_z \leftarrow c_0 + c_1] & l_x = c_0 \wedge l_y = c_1 \\ l[l_z \leftarrow \top] & \text{sonst} \end{cases}$

- ...

1. Letzte Woche
2. Motivation
3. Datenflussanalyse
4. Konstantenfaltung mit libFirm
5. Sonstiges

- TargetValue als Verband



- SSA-Form vereinfacht die Datenflussanalyse
 - Zu jedem Wert muss (global) nur ein TargetValue gespeichert werden

- Arbeitsliste zur effizienten Implementierung verwenden
 - Initial mit allen Knoten füllen
 - Wenn sich die Analyse-Information an einem Knoten geändert hat:
Alle Verwender des Knotens in die Arbeitsliste hängen

```
NodeCollector collector = new NodeCollector(worklist);  
graph.walkTopological(collector);
```

```
while (!worklist.isEmpty()) {  
    Node n = worklist.remove();  
    n.accept(this);  
}
```

Transformation des Graphen

```
for (Node node : latticeMap.keySet()) {  
    TargetValue tv = latticeMap.get(node);  
  
    if (tv.isConstant()) {  
        Node constant = graph.newConst(tv);  
        Graph.exchange(node, constant);  
    }  
}
```

1. Letzte Woche
2. Motivation
3. Datenflussanalyse
4. Konstantenfaltung mit libFirm
5. Sonstiges

Feedback! Fragen? Probleme?

- Anmerkungen?
- Probleme?
- Fragen?