6. Optimierung

Laufzeit und/oder Größe des Programms verringern bei unveränderter Wirkung.

Redundante Berechnungen eliminieren,

Berechnungen vereinfachen

Eingabe: Programm in Zwischensprache

Aufgaben: **Analyse** (Redundanz erkennen)
Transformationen durchführen

Ausgabe: verbessertes Programm in Zwischensprache

Analyse:

ermittelt statische Eigenschaften über Struktur und Ablauf des Programms (sicher, pessimistisch)

Analysekontext:

Ausdruck	lokal	
Grundblock	lokal	
Ablaufgraph (Prozedur)	global	intra-prozedural
Ablaufgraph u. Aufrufgraph	global	inter-prozedural

Transformation:

viele verschiedene Maßnahmen (U-53)

Vorbedingung für Anwendbarkeit (Analyse)

Wechselwirkung zwischen Transformationen

Vorlesung Übersetzer WS 97/98 / Folie 52

Ziele:

- Optimierungsanalyse und Transformationen verzahnt

in der Vorlesung:

- Übersicht

nachlesen:

Kastens / Übersetzerbau, Abschnitt 8

Verbessernde Transformationen

Liste einzelner Maßnahmen:

- **Algebraische Vereinfachung von Ausdrücken, Faltung**
  `2 * 3.14  x + 0  x * 2  x * * 2`
- **Konstantenweitergabe (constant propagation)**
  `x = 2; ... y = x * 5;`
- **Gemeinsame Teilausdrücke (common subexpressions)**
  `x = a * (b + c); ... y = (b + c)/2;`
- **Überflüssige Zuweisungen (dead variables)**
  `x = a + b; ... x = 5;`
- **Überflüssige Kopieranweisungen (copy propagation)**
  `x = y; ... ; z = x`
- **Nicht erreichbarer Code (dead code)**
  `b = true; ... if (b) x = 5; else y := 7;`
- **Code-Verschiebung (code motion)**
  `if (c) x = (a + b) * 2; else x = (a + b)/2;`
- **Einsetzen von Funktionsaufrufen (inlining)**
  `int Sqr (int i)  {return i * i;}`
- **Schleifen-invarianter Code**
  `while (b){... x = 5; ...}`
- **Operationen mit Schleifeninduktionsvariablen**
  `i = 1; while (b){k = i * 3; f(k); i = i + 1;}`

Zu jeder Transformation
Vorbedingung für sichere Anwendung
größere Chance für Anwendung, wenn Vorbedingung in größerem Kontext analysiert.

Wechselwirkungen:
Anwendung einer Transformation kann andere ermöglichen oder verhindern.

Vorlesung Übersetzer WS 97/98 / Folie 53

Ziele:

- Wichtige Transformationen an Beispielen kennenlernen

in der Vorlesung:

- Eläuterung der Vorbedingungen für einige Transformationen

nachlesen:

Kastens / Übersetzerbau, Abschnitt 8.1

Übungsaufgaben:

- Wenden Sie in gegebenen Beispielprogrammen möglichst viele der Transformationen an. Aufgabe 26

Verständnisfragen:

- Welche der Transformationen erfordern die Untersuchung von Programmpfaden?
- Geben Sie Paare von Transformationen an, so daß, erst nach Anwendung der ersten die zweite ermöglicht wird.

Analysemethoden - Übersicht

Struktur des Programms darstellen:

Ausdrucksbaum

gerichteter, azyklischer Ausdrucksgraph (DAG)

nach Erkennung gemeinsamer Teilausdrücke

Grundblock (basic block):

Folge von Anweisungen maximaler Länge, die alle ausgeführt werden, wenn die erste ausgeführt wird.

Ablaufgraph:

Knoten: Grundblöcke

Kanten: Verzweigungen zwischen Grundblöcken

Aufrufgraph:

Knoten: Funktionen

Kanten: `f->g` Funktion `f` enthält Aufruf von `g`

Analysen auf den Strukturen:

- **allgemeine Datenflußanalyse**
  z. B. Def-Use-Relation, Lebendigkeit für Variable, ...
- **Schleifen in beliebigem Ablaufgraph erkennen**
- **Induktionsvariable in Schleifen erkennen**
- **Analyse von Array-Indizes in Schleifen**
- **Aliasanalyse**

Vorlesung Übersetzer WS 97/98 / Folie 54

Ziele:

- Strukturen der Optimierungsanalyse kennenlernen

in der Vorlesung:

- Beispiele dazu

nachlesen:

Kastens / Übersetzerbau, Abschnitt 8.1, 8.1.3, 8.3.1

Übungsaufgaben:

- Zeichnen Sie zu vorgegebenen Programmen Ablaufgraphen und Aufrufgraphen.
  Aufgabe 27

Verständnisfragen:

- Wie hängen der Ablaufgraph und ein Ablauf des Programms zusammen?
- Unter welcher Bedingung kann man zwei Grundblöcke verschmelzen?
- Was bedeuten Kreise im Aufrufgraphen?

Datenflußgleichungen

Berechnung globaler Optimierungsinformation formuliert als Datenflußproblem über Ablaufgraph

z. B. „erreichende Definitionen“ (d: z := a):

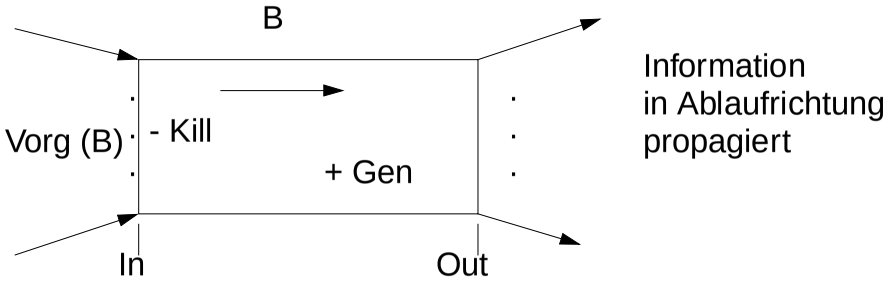
Welche Definitionen $\{d_i\}$ erreichen den Anfang von Block B? D. h. **es gibt einenPfad** von d_i nach B, auf dem nicht an die gleiche Variable zugewiesen wird.

Gleichungssystem für Vorwärtsprobleme

2 Gleichungen für jeden Block:

$$\text{Out}(B) = \text{Gen}(B) \cup (\text{In}(B) - \text{Kill}(B))$$

$$\text{In}(B) = \mathop{\Theta}_{h \in \text{Vorg}(B)} \text{Out}(h)$$



In, Out, Gen, Kill: Mengen von Objekten
z. B. Programmstellen, Variable, Ausdrücke

In, Out: Variablen des Gleichungssystems

Gen, Kill: Konstante für jeden Block

$\Theta = \cup$ **Existenzproblem** (es gibt einen Pfad); **minimale** Lösung

$\Theta = \cap$ **Allproblem** (für alle Pfade); **maximale** Lösung

Vorlesung Übersetzer WS 97/98 / Folie 55

Ziele:

- Optimierungsprobleme durch Gleichungssysteme modellieren

in der Vorlesung:

- am Beispiel "erreichende Definitionen" erläutern (siehe U-55a)

nachlesen:

Kastens / Übersetzerbau, Abschnitt 8.2.4

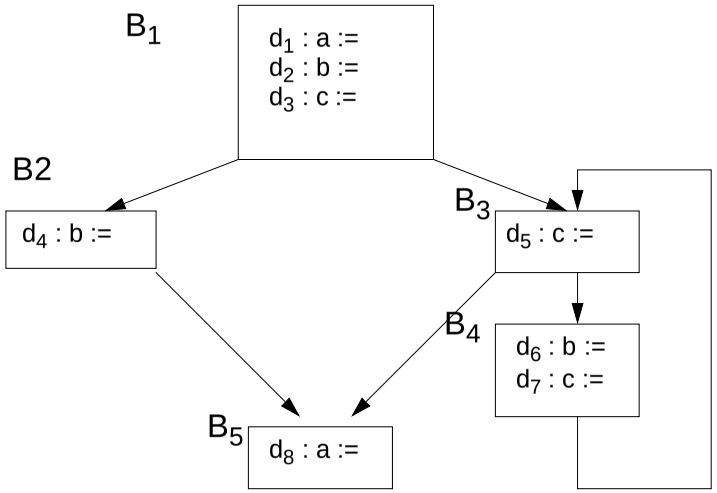
Übungsaufgaben:

- Gleichungssystem für ein gegebenes Programm aufstellen und lösen.

Verständnisfragen:

- Wie wird die Optimierungsinformation für Programmstellen innerhalb eines Grundblockes errechnet?
- Begründen Sie: Existenzproblem - minimale Lösung, Allproblem - maximale Lösung.

Beispiel: Erreichende Definitionen



	Gen	Kill	In	Out
B_1	d_1, d_2, d_3	d_4, d_5, d_6, d_7, d_8	$\varnothing$	d_1, d_2, d_3
B_2	d_4	d_2, d_6	d_1, d_2, d_3	d_1, d_3, d_4
B_3	d_5	d_3, d_7	d_1, d_2, d_3, d_6, d_7	d_1, d_2, d_5, d_6
B_4	d_6, d_7	d_2, d_3, d_4, d_5	d_1, d_2, d_5, d_6	d_1, d_6, d_7
B_5	d_8	d_1	$d_1, d_2, d_3, d_4, d_5, d_6$	$d_2, d_3, d_4, d_5, d_6, d_8$

Gen (B): Definition d_i: v := A in Block B, auf die in B keine weitere Definition für v folgt.

Kill (B): alle Definitionen d_j: v := A nicht in B, an deren Variable auch in B zugewiesen wird.

Vorlesung Übersetzer WS 97/98 / Folie 55a

Ziele:

Beispiel zu U-55

nachlesen:

Kastens / Übersetzerbau, Abschnitt 8.2.4, Abb. 8.2-5

Rückwärtsprobleme

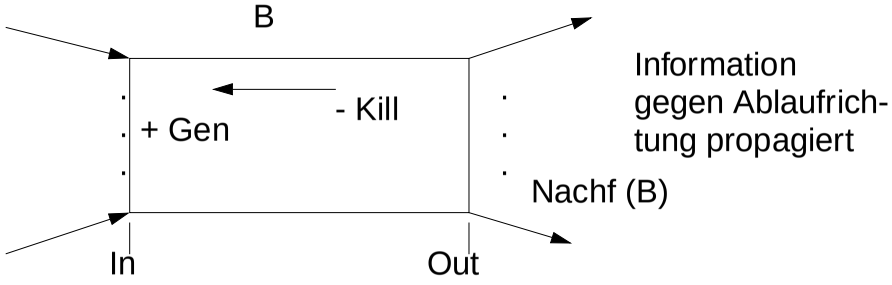
z. B. „lebendige Variable“:

Welche Variablen $\{v_i\}$ sind am Ende von Block B lebendig? D. h. **es gibt einen Pfad** von B zu einer Benutzung von v_i, auf dem nicht an v_i zugewiesen wird.

Gleichungsschema für Rückwärtsprobleme:

$$In(B) = Gen(B) \cup (Out(B) - Kill(B))$$

$$Out(B) = \overset{\Theta}{\underset{h \in Nachf(B)}{=}} In(h)$$



„lebendige Variable“: $\Theta = \cup$

Gen (B): Variable, die in B benutzt, und vor der Benutzung in B nicht definiert werden

Kill (B): Variable, die in B definiert, und vor der Definition in B nicht benutzt werden

Existenz- und **Allprobleme** wie bei Vorwärtsproblemen

Vorlesung Übersetzer WS 97/98 / Folie 56

Ziele:

- analoges Gleichungsschema für Rückwärtsanalyse

in der Vorlesung:

- Anwendung des Beispiels "lebendige Variablen"

nachlesen:

Kastens / Übersetzerbau, Abschnitt 8.3.3

Übungsaufgaben:

- Gleichungsschema für "lebendige Variablen" zu einen Programm aufstellen und lösen. Aufgabe 28

Verständnisfragen:

- Wie wird die Optimierungsinformation für Programmstellen innerhalb eines Grundblockes errechnet?

Iterative Lösung von Datenflußgleichungen

Initialisierung Existenzproblem:

```
for all B do begin In(B):=∅; Out(B):=Gen(B) end;
```

Initialisierung Allproblem:

```
for all B do
   begin In(B) := U; {volle Menge}
         Out(B):= Gen(B) ∪ (U-Kill(B))
   end;
repeat
   stable := true;
   for all B ≠ Ba   {*}
   do begin
      for all V ∈ Vorg(B) do
         In(B):= In(B) Θ Out(V);
         oldout:= Out(B);
         Out(B):= Gen(B) ∪ (In(B)-Kill(B));
         stable:= stable and Out(B)=oldout
   end
until stable
```

Komplexität: $0(n^3)$ n = Anzahl Grundblöcke

bzw. $0(n^2)$ falls $|Vorg(B)| \leq k << n$

heuristische Verbesserung:

Änderungen möglichst schnell propagieren. Deshalb Blöcke ohne Rückwärtskanten topologisch sortieren und bei {*} in dieser Ordnung durchlaufen. Dann wird die Anzahl der repeat-Durchläufe nur durch die Zahl der Rückwärtskanten bestimmt.

Bei Rückwärtsproblem entsprechend mit umgekehrter Kantenrichtung sortieren.

hierarchisches Verfahren Intervallanalyse:
prägt dem Ablaufgraphen Struktur auf; berechnet Gen, Kill aufwärts, In, Out abwärts in der Hierarchie.

Vorlesung Übersetzer WS 97/98 / Folie 57

Ziele:

- Einfachen, iterativen Lösungsalgorithmus kennenlernen

in der Vorlesung:

- Algorithmus erläutern, Terminierung, Komplexität

nachlesen:

Kastens / Übersetzerbau, Abschnitt 8.2.5 (weiterführend 8.2.6)

Verständnisfragen:

- Begründen Sie, warum die Bearbeitung der Blöcke in der angegebenen Reihenfolge eine Verbesserung bewirkt.

Schleifenerkennung im Ablaufgraphen

Schleifen gebildet durch Sprünge, aus Quellprogramm oder im Zwischen-Code erzeugt

Relation über Knoten des Ablaufgraphen:

a **dominiert** b: (reflexiv: a dominiert a):

Jeder Weg vom Anfangsknoten zu b führt über a.

Schleifeneigenschaft:

- **eindeutiger Eingangsknoten h** dominiert alle Knoten der Schleife, h ist Schleifenkopf
- **von jedem Knoten der Schleife führt ein Weg zu h**

Rückwärtskante im Ablaufgraph n -> d mit d dominiert n

Schleife zur Rückwärtskante n -> d:

d und alle Knoten, die n erreichen, ohne d zu berühren

d ist der Eingangsknoten

$S = \{d, n\} \cup \{ v \mid v = Vorg(a), a \in S, a \neq d\}$

Vorlesung Übersetzer WS 97/98 / Folie 58

Ziele:

- Bedeutung der Dominatorrelation zur Strukturerkennung in gerichteten Graphen

in der Vorlesung:

- Erläuterung am Beispiel (U-58a)

nachlesen:

Kastens / Übersetzerbau, Abschnitt 8.2.2

Übungsaufgaben:

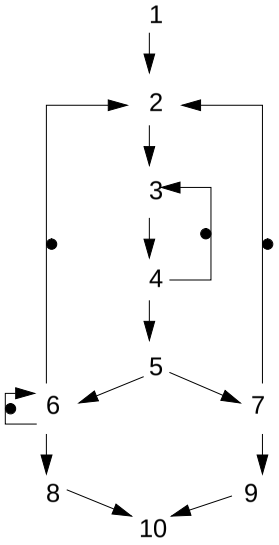
- Zu gegebenem Graph Schleifen bestimmen und klassifizieren. Aufgabe 28

Verständnisfragen:

- Geben Sie Graphen an, die Kreise aber keine Schleifen enthalten.
- Aus welchen Programmstrukturen entstehen solche Graphen?

Rückwärtskanten und Schleifen im Ablaufgraph

Beispiel:



Rückwärtskante:	Schleife:
4 -> 3	S_1 = {3,4}
6 -> 2	S_2 = {2, 3, 4, 5, 6}
7 -> 2	S_3 = {2, 3, 4, 5, 7}
6 -> 6	S_4 = {6}

Schleifen sind

- **disjunkt** $S_1 \cap S_4 = \varnothing$
- **geschachtelt** $S_1 \subset S_2$
- **nicht geschachtelt, aber gleicher Kopf** S_2, S_3
  (zu einer Schleife vereinigen)

Vorlesung Übersetzer WS 97/98 / Folie 58a

Ziele:

- Beispiel zur Schleifenbestimmung (U-58)

in der Vorlesung:

- Konstruktionsverfahren zeigen

nachlesen:

Kastens / Übersetzerbau, Abschnitt 8.2.2, Abb. 8.2-2

Verständnisfragen:

- Geben Sie ein Programm zu dem Ablaufgraphen an.