



OptComp

A. Koch

Organisatorisch

Einleitung

SSA
Erzeugen

SSA für
strukturierte
Sprachen

Dominatoren

Rückwandlung
aus
SSA-Form

Optimierende Compiler

7. Static Single Assignment-Form

Andreas Koch

FG Eingebettete Systeme und ihre Anwendungen
Informatik, TU Darmstadt

Sommersemester 2006



- Kurze Unterbrechung zur Datenflußanalyse
 - Stoff von letzter Woche (globale Redundanzeliminierung)
- Stoff der nächsten Aufgabe
 - Aber nur noch auf Untermenge von Triangle
 - Nur noch Hauptprogramm
 - Nur noch skalare Variablen
- Warum?
 - Behandlung aller Sonderfälle sehr aufwendig
 - ... bei minimalem Lerneffekt
- Bei Interesse
 - Alle Details in Papers nachlesen

OptComp

A. Koch

Organisatorisch

Einleitung

SSA
Erzeugen

SSA für
strukturierte
Sprachen

Dominatoren

Rückwandlung
aus
SSA-Form



- Kurze Unterbrechung zur Datenflußanalyse
 - Stoff von letzter Woche (globale Redundanzeliminierung)
- Stoff der nächsten Aufgabe
 - Aber nur noch auf Untermenge von Triangle
 - Nur noch Hauptprogramm
 - Nur noch skalare Variablen
- Warum?
 - Behandlung aller Sonderfälle sehr aufwendig
 - ... bei minimalem Lerneffekt
- Bei Interesse
 - Alle Details in Papers nachlesen

OptComp

A. Koch

Organisatorisch

Einleitung

SSA
Erzeugen

SSA für
strukturierte
Sprachen

Dominatoren

Rückwandlung
aus
SSA-Form



- Kurze Unterbrechung zur Datenflußanalyse
 - Stoff von letzter Woche (globale Redundanzeliminierung)
- Stoff der nächsten Aufgabe
 - Aber nur noch auf Untermenge von Triangle
 - Nur noch Hauptprogramm
 - Nur noch skalare Variablen
- Warum?
 - Behandlung aller Sonderfälle sehr aufwendig
 - ... bei minimalem Lerneffekt
- Bei Interesse
 - Alle Details in Papers nachlesen

OptComp

A. Koch

Organisatorisch

Einleitung

SSA

Erzeugen

SSA für
strukturierte
Sprachen

Dominatoren

Rückwandlung
aus
SSA-Form



- Kurze Unterbrechung zur Datenflußanalyse
 - Stoff von letzter Woche (globale Redundanzeliminierung)
- Stoff der nächsten Aufgabe
 - Aber nur noch auf Untermenge von Triangle
 - Nur noch Hauptprogramm
 - Nur noch skalare Variablen
- Warum?
 - Behandlung aller Sonderfälle sehr aufwendig
 - ... bei minimalem Lerneffekt
- Bei Interesse
 - Alle Details in Papers nachlesen

OptComp

A. Koch

Organisatorisch

Einleitung

SSA
Erzeugen

SSA für
strukturierte
Sprachen

Dominatoren

Rückwandlung
aus
SSA-Form



- Kurze Unterbrechung zur Datenflußanalyse
 - Stoff von letzter Woche (globale Redundanzeliminierung)
- Stoff der nächsten Aufgabe
 - Aber nur noch auf Untermenge von Triangle
 - Nur noch Hauptprogramm
 - Nur noch skalare Variablen
- Warum?
 - Behandlung aller Sonderfälle sehr aufwendig
 - ... bei minimalem Lerneffekt
- Bei Interesse
 - Alle Details in Papers nachlesen

OptComp

A. Koch

Organisatorisch

Einleitung

SSA
Erzeugen

SSA für
strukturierte
Sprachen

Dominatoren

Rückwandlung
aus
SSA-Form



- Kurze Unterbrechung zur Datenflußanalyse
 - Stoff von letzter Woche (globale Redundanzeliminierung)
- Stoff der nächsten Aufgabe
 - Aber nur noch auf Untermenge von Triangle
 - Nur noch Hauptprogramm
 - Nur noch skalare Variablen
- Warum?
 - Behandlung aller Sonderfälle sehr aufwendig
 - ... bei minimalem Lerneffekt
- Bei Interesse
 - Alle Details in Papers nachlesen

OptComp

A. Koch

Organisatorisch

Einleitung

SSA
Erzeugen

SSA für
strukturierte
Sprachen

Dominatoren

Rückwandlung
aus
SSA-Form



- Kurze Unterbrechung zur Datenflußanalyse
 - Stoff von letzter Woche (globale Redundanzeliminierung)
- Stoff der nächsten Aufgabe
 - Aber nur noch auf Untermenge von Triangle
 - Nur noch Hauptprogramm
 - Nur noch skalare Variablen
- Warum?
 - Behandlung aller Sonderfälle sehr aufwendig
 - ... bei minimalem Lerneffekt
- Bei Interesse
 - Alle Details in Papers nachlesen

OptComp

A. Koch

Organisatorisch

Einleitung

SSA
Erzeugen

SSA für
strukturierte
Sprachen

Dominatoren

Rückwandlung
aus
SSA-Form



- Kurze Unterbrechung zur Datenflußanalyse
 - Stoff von letzter Woche (globale Redundanzeliminierung)
- Stoff der nächsten Aufgabe
 - Aber nur noch auf Untermenge von Triangle
 - Nur noch Hauptprogramm
 - Nur noch skalare Variablen
- Warum?
 - Behandlung aller Sonderfälle sehr aufwendig
 - ... bei minimalem Lerneffekt
- Bei Interesse
 - Alle Details in Papers nachlesen

OptComp

A. Koch

Organisatorisch

Einleitung

SSA
Erzeugen

SSA für
strukturierte
Sprachen

Dominatoren

Rückwandlung
aus
SSA-Form



- Kurze Unterbrechung zur Datenflußanalyse
 - Stoff von letzter Woche (globale Redundanzeliminierung)
- Stoff der nächsten Aufgabe
 - Aber nur noch auf Untermenge von Triangle
 - Nur noch Hauptprogramm
 - Nur noch skalare Variablen
- Warum?
 - Behandlung aller Sonderfälle sehr aufwendig
 - ... bei minimalem Lerneffekt
- Bei Interesse
 - Alle Details in Papers nachlesen

OptComp

A. Koch

Organisatorisch

Einleitung

SSA
Erzeugen

SSA für
strukturierte
Sprachen

Dominatoren

Rückwandlung
aus
SSA-Form



- Kurze Unterbrechung zur Datenflußanalyse
 - Stoff von letzter Woche (globale Redundanzeliminierung)
- Stoff der nächsten Aufgabe
 - Aber nur noch auf Untermenge von Triangle
 - Nur noch Hauptprogramm
 - Nur noch skalare Variablen
- Warum?
 - Behandlung aller Sonderfälle sehr aufwendig
 - ... bei minimalem Lerneffekt
- Bei Interesse
 - Alle Details in Papers nachlesen

OptComp

A. Koch

Organisatorisch

Einleitung

SSA
Erzeugen

SSA für
strukturierte
Sprachen

Dominatoren

Rückwandlung
aus
SSA-Form



- Kurze Unterbrechung zur Datenflußanalyse
 - Stoff von letzter Woche (globale Redundanzeliminierung)
- Stoff der nächsten Aufgabe
 - Aber nur noch auf Untermenge von Triangle
 - Nur noch Hauptprogramm
 - Nur noch skalare Variablen
- Warum?
 - Behandlung aller Sonderfälle sehr aufwendig
 - ... bei minimalem Lerneffekt
- Bei Interesse
 - **Alle** Details in Papers nachlesen

OptComp

A. Koch

Organisatorisch

Einleitung

SSA
Erzeugen

SSA für
strukturierte
Sprachen

Dominatoren

Rückwandlung
aus
SSA-Form



Ab jetzt Auszüge aus:

Single-Pass Generation of Static Single Assignment Form
for Structured Languages

MARC M. BRANDIS and HANSPETER MÖSSENBÖCK

ACM Transactions on Programming Languages and
Systems 16(6): 1684-1698, Nov.1994

- Erzeugung von SSA-Form aus strukturierten Programmierprachen
- Sehr gut zu lesen

OptComp

A. Koch

Organisatorisch

Einleitung

SSA
Erzeugen

SSA für
strukturierte
Sprachen

Dominatoren

Rückwandlung
aus
SSA-Form



Ab jetzt Auszüge aus:

Single-Pass Generation of Static Single Assignment Form
for Structured Languages

MARC M. BRANDIS and HANSPETER MÖSSENBÖCK

ACM Transactions on Programming Languages and
Systems 16(6): 1684-1698, Nov.1994

- Erzeugung von SSA-Form aus strukturierten Programmierprachen
- Sehr gut zu lesen

OptComp

A. Koch

Organisatorisch

Einleitung

SSA
Erzeugen

SSA für
strukturierte
Sprachen

Dominatoren

Rückwandlung
aus
SSA-Form



Practical Improvements to the Construction and Destruction of Static Single Assignment Form

BRIGGS, COOPER, HARVEY, SIMPSON

SOFTWARE: PRACTICE AND EXPERIENCE, VOL. 28(8),
128 (July 1998)

- Umwandeln aus der SSA-Form (→ nächste Woche)
- Recht gut zu lesen

OptComp

A. Koch

Organisatorisch

Einleitung

SSA
Erzeugen

SSA für
strukturierte
Sprachen

Dominatoren

Rückwandlung
aus
SSA-Form



Practical Improvements to the Construction and Destruction of Static Single Assignment Form

BRIGGS, COOPER, HARVEY, SIMPSON

SOFTWARE: PRACTICE AND EXPERIENCE, VOL. 28(8),
128 (July 1998)

- Umwandeln aus der SSA-Form (→ nächste Woche)
- Recht gut zu lesen

OptComp

A. Koch

Organisatorisch

Einleitung

SSA
Erzeugen

SSA für
strukturierte
Sprachen

Dominatoren

Rückwandlung
aus
SSA-Form



Efficiently computing static single assignment form and the control dependence graph

CYTRON, FERRANTE, ROSEN, WEGMAN, ZADECK

ACM Transactions on Programming Languages and Systems (TOPLAS), Volume 13 , Issue 4 (October 1991)

- *Das SSA-Paper schlechthin*
- Keine ganz einfache Lektüre mehr
- Aber viele Details nur hier behandelt

OptComp

A. Koch

Organisatorisch

Einleitung

SSA
Erzeugen

SSA für
strukturierte
Sprachen

Dominatoren

Rückwandlung
aus
SSA-Form



Efficiently computing static single assignment form and the control dependence graph

CYTRON, FERRANTE, ROSEN, WEGMAN, ZADECK

ACM Transactions on Programming Languages and Systems (TOPLAS), Volume 13 , Issue 4 (October 1991)

- *Das SSA-Paper schlechthin*
- Keine ganz einfache Lektüre mehr
- Aber viele Details nur hier behandelt

OptComp

A. Koch

Organisatorisch

Einleitung

SSA
Erzeugen

SSA für
strukturierte
Sprachen

Dominatoren

Rückwandlung
aus
SSA-Form



Efficiently computing static single assignment form and the control dependence graph

CYTRON, FERRANTE, ROSEN, WEGMAN, ZADECK

ACM Transactions on Programming Languages and Systems (TOPLAS), Volume 13 , Issue 4 (October 1991)

- *Das SSA-Paper schlechthin*
- Keine ganz einfache Lektüre mehr
- Aber viele Details nur hier behandelt

OptComp

A. Koch

Organisatorisch

Einleitung

SSA
Erzeugen

SSA für
strukturierte
Sprachen

Dominatoren

Rückwandlung
aus
SSA-Form

Static Single Assignment-Form



OptComp

A. Koch

Organisatorisch

Einleitung

SSA
Erzeugen

SSA für
strukturierte
Sprachen

Dominatoren

Rückwandlung
aus
SSA-Form

```
// Normal                // SSA-Form
v := 0;                  v1 := 0;
x := v + 1;              x1 := v1 + 1;
v := 2;                  v2 := 2;
y := v + 3                y1 := v2 + 3
```

- Nur noch genau eine Zuweisung an jede Variable
 - Erzeuge eindeutige Namen für gleiche Zuweisungsziele
 - Numerierte Variablen sind Wertinstanzen der ursprünglichen Variablen
 - Kurz als Werte bezeichnet

Static Single Assignment-Form



OptComp

A. Koch

Organisatorisch

Einleitung

SSA
Erzeugen

SSA für
strukturierte
Sprachen

Dominatoren

Rückwandlung
aus
SSA-Form

```
// Normal                // SSA-Form
v := 0;                  v1 := 0;
x := v + 1;              x1 := v1 + 1;
v := 2;                  v2 := 2;
y := v + 3;              y1 := v2 + 3
```

- Nur noch genau eine Zuweisung an jede Variable
 - Erzeuge eindeutige Namen für gleiche Zuweisungsziele
 - Numerierte Variablen sind Wertinstanzen der ursprünglichen Variablen
 - Kurz als Werte bezeichnet



```
// Normal                // SSA-Form
v := 0;                  v1 := 0;
x := v + 1;              x1 := v1 + 1;
v := 2;                  v2 := 2;
y := v + 3                y1 := v2 + 3
```

- Nur noch genau eine Zuweisung an jede Variable
 - Erzeuge eindeutige Namen für gleiche Zuweisungsziele
 - Numerierte Variablen sind **Wertinstanzen** der ursprünglichen Variablen
 - Kurz als **Werte** bezeichnet



<pre>// Normal v := 0; x := v + 1; v := 2; y := v + 3</pre>	<pre>// SSA-Form v1 := 0; x1 := v1 + 1; v2 := 2; y1 := v2 + 3</pre>
---	---

- Nur noch genau eine Zuweisung an jede Variable
 - Erzeuge eindeutige Namen für gleiche Zuweisungsziele
 - Nummerierte Variablen sind **Wertinstanzen** der ursprünglichen Variablen
 - Kurz als **Werte** bezeichnet

Problem: Kontrollfluß



Was passiert, wenn zwei Werte der gleichen Variable aufeinanderstoßen?

- An sogenanntem *merge* oder *join*-Punkten im Kontrollflußgraphen

➔ Auflösung über Phi-Funktion

- Für jeden Kontrollzweig einen Parameter
- Liefert als Ergebnis den Wert entsprechend der genommenen Kante

OptComp

A. Koch

Organisatorisch

Einleitung

SSA
Erzeugen

SSA für
strukturierte
Sprachen

Dominatoren

Rückwandlung
aus
SSA-Form



Was passiert, wenn zwei Werte der gleichen Variable aufeinanderstoßen?

- An sogenanntem *merge* oder *join*-Punkten im Kontrollflußgraphen

➔ Auflösung über Phi-Funktion

- Für jeden Kontrollzweig einen Parameter
- Liefert als Ergebnis den Wert entsprechend der genommenen Kante

OptComp

A. Koch

Organisatorisch

Einleitung

SSA
Erzeugen

SSA für
strukturierte
Sprachen

Dominatoren

Rückwandlung
aus
SSA-Form



Was passiert, wenn zwei Werte der gleichen Variable aufeinanderstoßen?

- An sogenanntem *merge* oder *join*-Punkten im Kontrollflußgraphen

➔ Auflösung über Phi-Funktion

- Für jeden Kontrollzweig einen Parameter
- Liefert als Ergebnis den Wert entsprechend der genommenen Kante

OptComp

A. Koch

Organisatorisch

Einleitung

SSA
Erzeugen

SSA für
strukturierte
Sprachen

Dominatoren

Rückwandlung
aus
SSA-Form

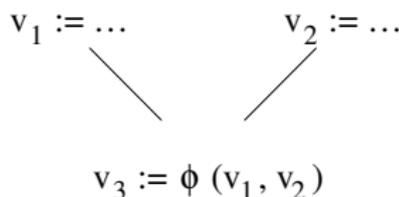
Problem: Kontrollfluß



Was passiert, wenn zwei Werte der gleichen Variable aufeinanderstoßen?

- An sogenanntem *merge* oder *join*-Punkten im Kontrollflußgraphen

➔ Auflösung über Phi-Funktion



- Für jeden Kontrollzweig einen Parameter
 - Den jeweiligen Wert
- Liefert als Ergebnis den Wert entsprechend der genommenen Kante
 - Von welchem Zweig kamen wir?
 - Welcher Wert ist also der richtige?

OptComp

A. Koch

Organisatorisch

Einleitung

SSA
Erzeugen

SSA für
strukturierte
Sprachen

Dominatoren

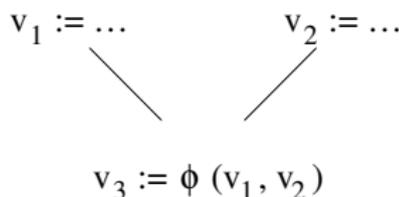
Rückwandlung
aus
SSA-Form



Was passiert, wenn zwei Werte der gleichen Variable aufeinanderstoßen?

- An sogenanntem *merge* oder *join*-Punkten im Kontrollflußgraphen

➔ Auflösung über Phi-Funktion



- Für jeden Kontrollzweig einen Parameter
 - Den jeweiligen Wert
- Liefert als Ergebnis den Wert entsprechend der genommenen Kante
 - Von welchem Zweig kamen wir?
 - Welcher Wert ist also der richtige?

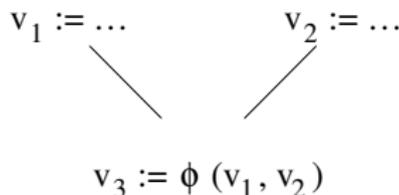
Problem: Kontrollfluß



Was passiert, wenn zwei Werte der gleichen Variable aufeinanderstoßen?

- An sogenanntem *merge* oder *join*-Punkten im Kontrollflußgraphen

➔ Auflösung über Phi-Funktion



- Für jeden Kontrollzweig einen Parameter
 - Den jeweiligen Wert
- Liefert als Ergebnis den Wert entsprechend der genommenen Kante
 - Von welchem Zweig kamen wir?
 - Welcher Wert ist also der richtige?

OptComp

A. Koch

Organisatorisch

Einleitung

SSA
Erzeugen

SSA für
strukturierte
Sprachen

Dominatoren

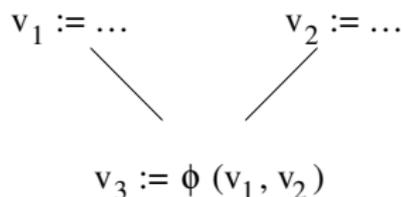
Rückwandlung
aus
SSA-Form



Was passiert, wenn zwei Werte der gleichen Variable aufeinanderstoßen?

- An sogenanntem *merge* oder *join*-Punkten im Kontrollflußgraphen

➔ Auflösung über Phi-Funktion



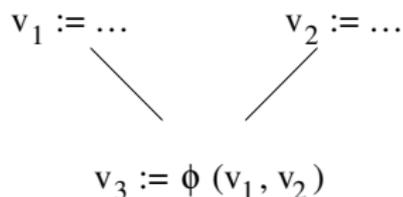
- Für jeden Kontrollzweig einen Parameter
 - Den jeweiligen Wert
- Liefert als Ergebnis den Wert entsprechend der genommenen Kante
 - Von welchem Zweig kamen wir?
 - Welcher Wert ist also der richtige?



Was passiert, wenn zwei Werte der gleichen Variable aufeinanderstoßen?

- An sogenanntem *merge* oder *join*-Punkten im Kontrollflußgraphen

➔ Auflösung über Phi-Funktion



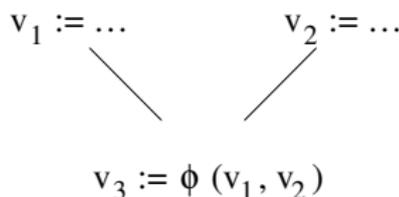
- Für jeden Kontrollzweig einen Parameter
 - Den jeweiligen Wert
- Liefert als Ergebnis den Wert entsprechend der genommenen Kante
 - Von welchem Zweig kamen wir?
 - Welcher Wert ist also der richtige?



Was passiert, wenn zwei Werte der gleichen Variable aufeinanderstoßen?

- An sogenanntem *merge* oder *join*-Punkten im Kontrollflußgraphen

➔ Auflösung über Phi-Funktion

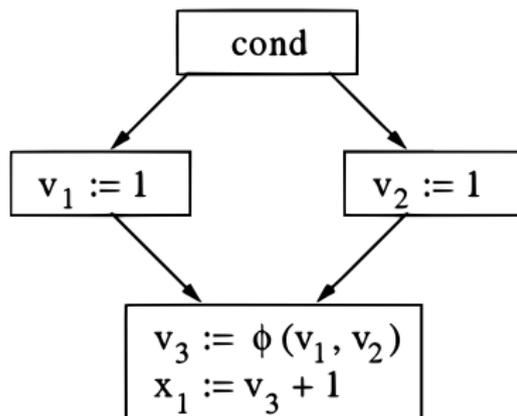


- Für jeden Kontrollzweig einen Parameter
 - Den jeweiligen Wert
- Liefert als Ergebnis den Wert entsprechend der genommenen Kante
 - Von welchem Zweig kamen wir?
 - Welcher Wert ist also der richtige?

Beispiel SSA-Form: IF-Statement



```
IF cond THEN
  v := 1
ELSE
  v := 2
END;
x := v + 1
```



OptComp

A. Koch

Organisatorisch

Einleitung

SSA
Erzeugen

SSA für
strukturierte
Sprachen

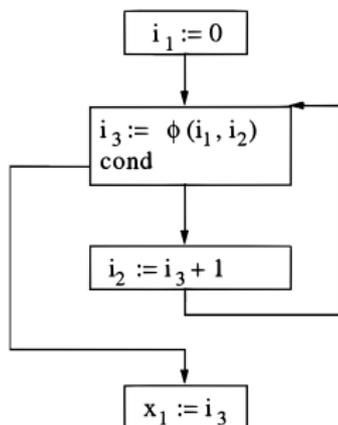
Dominatoren

Rückwandlung
aus
SSA-Form

Beispiel SSA-Form: WHILE-Statement



```
i := 0;  
WHILE cond DO  
  i := i + 1  
END;  
x := i
```

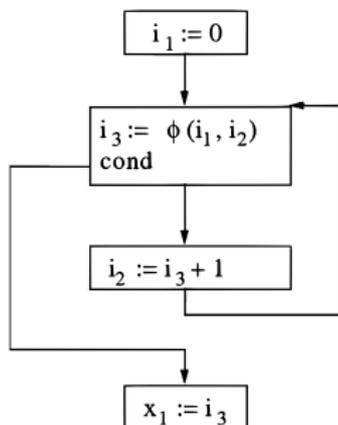


Beachte: Entscheidung, ob Wert von **vor** oder **nach** dem Schleifenkörper genommen wird.

Beispiel SSA-Form: WHILE-Statement



```
i := 0;  
WHILE cond DO  
  i := i + 1  
END;  
x := i
```



Beachte: Entscheidung, ob Wert von **vor** oder **nach** dem Schleifenkörper genommen wird.



- Für jeden Wert genau eine Definition
- Jede Zuweisung legt neuen Wert an
- Kein Auslöschen (*kill*) von Werten möglich
- Wenn zwei Ausdrücke textuell gleich sind
- ... liefern sie das gleiche Ergebnis



OptComp

A. Koch

Organisatorisch

Einleitung

SSA
Erzeugen

SSA für
strukturierte
Sprachen

Dominatoren

Rückwandlung
aus
SSA-Form

- Für jeden Wert genau eine Definition
- Jede Zuweisung legt neuen Wert an
- Kein Auslöschen (*kill*) von Werten möglich
- Wenn zwei Ausdrücke textuell gleich sind
- ... liefern sie das gleiche Ergebnis



OptComp

A. Koch

Organisatorisch

Einleitung

SSA
Erzeugen

SSA für
strukturierte
Sprachen

Dominatoren

Rückwandlung
aus
SSA-Form

- Für jeden Wert genau eine Definition
- Jede Zuweisung legt neuen Wert an
- Kein Auslöschen (*kill*) von Werten möglich
- Wenn zwei Ausdrücke textuell gleich sind
- ... liefern sie das gleiche Ergebnis



OptComp

A. Koch

Organisatorisch

Einleitung

SSA
Erzeugen

SSA für
strukturierte
Sprachen

Dominatoren

Rückwandlung
aus
SSA-Form

- Für jeden Wert genau eine Definition
- Jede Zuweisung legt neuen Wert an
- Kein Auslöschen (*kill*) von Werten möglich
- Wenn zwei Ausdrücke textuell gleich sind
- ... liefern sie das gleiche Ergebnis



- Für jeden Wert genau eine Definition
- Jede Zuweisung legt neuen Wert an
- Kein Auslöschen (*kill*) von Werten möglich
- Wenn zwei Ausdrücke textuell gleich sind
- ... liefern sie das gleiche Ergebnis



Drei Teilprobleme

- 1 Eindeutige Namen für Werte
 - Einfach durchnummerieren
- 2 Einfügen von Phi-Funktionen
 - Holzhammermethode
 - Erzeugt **sehr viele** Phi-Funktionen, die meisten unnötig
- 3 Umbenennen von benutzten Variablen in passende Werte
 - Wieder recht einfach
 - Referenziert letzte Definition



Drei Teilprobleme

- 1 Eindeutige Namen für Werte
 - Einfach durchnummerieren
- 2 Einfügen von Phi-Funktionen
 - Holzhammermethode
 - Erzeugt **sehr viele** Phi-Funktionen, die meisten unnötig
- 3 Umbenennen von benutzten Variablen in passende Werte
 - Wieder recht einfach
 - Referenziert letzte Definition



Drei Teilprobleme

- 1 Eindeutige Namen für Werte
 - Einfach durchnummerieren
- 2 Einfügen von Phi-Funktionen
 - Holzhammermethode
 - An jedem join-Point für **alle** Variablen Phi-Funktionen einfügen
 - Erzeugt **sehr viele** Phi-Funktionen, die meisten unnötig
- 3 Umbenennen von benutzten Variablen in passende Werte
 - Wieder recht einfach
 - Referenziert letzte Definition



Drei Teilprobleme

- 1 Eindeutige Namen für Werte
 - Einfach durchnummerieren
- 2 Einfügen von Phi-Funktionen
 - Holzhammermethode
 - An jedem join-Point für **alle** Variablen Phi-Funktionen einfügen
 - Erzeugt **sehr viele** Phi-Funktionen, die meisten unnötig
- 3 Umbenennen von benutzten Variablen in passende Werte
 - Wieder recht einfach
 - Referenziert letzte Definition



Drei Teilprobleme

- 1 Eindeutige Namen für Werte
 - Einfach durchnummerieren
- 2 Einfügen von Phi-Funktionen
 - Holzhammermethode
 - An jedem join-Point für **alle** Variablen Phi-Funktionen einfügen
 - Erzeugt **sehr viele** Phi-Funktionen, die meisten unnötig
- 3 Umbenennen von benutzten Variablen in passende Werte
 - Wieder recht einfach
 - Referenziert letzte Definition



Drei Teilprobleme

- 1 Eindeutige Namen für Werte
 - Einfach durchnummerieren
- 2 Einfügen von Phi-Funktionen
 - Holzhammermethode
 - An jedem join-Point für **alle** Variablen Phi-Funktionen einfügen
 - Erzeugt **sehr viele** Phi-Funktionen, die meisten unnötig
- 3 Umbenennen von benutzten Variablen in passende Werte
 - Wieder recht einfach
 - Referenziert letzte Definition



Drei Teilprobleme

- 1 Eindeutige Namen für Werte
 - Einfach durchnummerieren
- 2 Einfügen von Phi-Funktionen
 - Holzhammermethode
 - An jedem join-Point für **alle** Variablen Phi-Funktionen einfügen
 - Erzeugt **sehr viele** Phi-Funktionen, die meisten unnötig
- 3 Umbenennen von benutzten Variablen in passende Werte
 - Wieder recht einfach
 - Referenziert letzte Definition



Drei Teilprobleme

- 1 Eindeutige Namen für Werte
 - Einfach durchnummerieren
- 2 Einfügen von Phi-Funktionen
 - Holzhammermethode
 - An jedem join-Point für **alle** Variablen Phi-Funktionen einfügen
 - Erzeugt **sehr viele** Phi-Funktionen, die meisten unnötig
- 3 Umbenennen von benutzten Variablen in passende Werte
 - Wieder recht einfach
 - Referenziert letzte Definition



Drei Teilprobleme

- 1 Eindeutige Namen für Werte
 - Einfach durchnummerieren
- 2 Einfügen von Phi-Funktionen
 - Holzhammermethode
 - An jedem join-Point für **alle** Variablen Phi-Funktionen einfügen
 - Erzeugt **sehr viele** Phi-Funktionen, die meisten unnötig
- 3 Umbenennen von benutzten Variablen in passende Werte
 - Wieder recht einfach
 - Referenziert letzte Definition



Allgemeine Lösung

- Cytron et. al. 1991
- Vorgehen: Berechnen von Dominatorgrenzen
- “Gerade nicht mehr” von Knoten X dominierte Knoten
- Hier nicht mehr klar, ob Definitionen aus X noch gelten
- Einfügen von Phi-Knoten nur für die Variablen, bei denen entschieden werden muß
 - Aufeinandertreffen von verschiedenen Definitionen an Dominatorgrenzen
- Algorithmus nicht trivial ...

OptComp

A. Koch

Organisatorisch

Einleitung

SSA
Erzeugen

SSA für
strukturierte
Sprachen

Dominatoren

Rückwandlung
aus
SSA-Form



Allgemeine Lösung

- Cytron et. al. 1991
- Vorgehen: Berechnen von Dominatorgrenzen
- “Gerade nicht mehr” von Knoten X dominierte Knoten
- Hier nicht mehr klar, ob Definitionen aus X noch gelten
- Einfügen von Phi-Knoten nur für die Variablen, bei denen entschieden werden muß
 - Aufeinandertreffen von verschiedenen Definitionen an Dominatorgrenzen
- Algorithmus nicht trivial ...



Allgemeine Lösung

- Cytron et. al. 1991
- Vorgehen: Berechnen von Dominatorgrenzen
- “Gerade nicht mehr” von Knoten X dominierte Knoten
- Hier nicht mehr klar, ob Definitionen aus X noch gelten
- Einfügen von Phi-Knoten nur für die Variablen, bei denen entschieden werden muß
 - Aufeinandertreffen von verschiedenen Definitionen an Dominatorgrenzen
- Algorithmus nicht trivial ...



Allgemeine Lösung

- Cytron et. al. 1991
- Vorgehen: Berechnen von Dominatorgrenzen
- “Gerade nicht mehr” von Knoten X dominierte Knoten
- Hier nicht mehr klar, ob Definitionen aus X noch gelten
- Einfügen von Phi-Knoten nur für die Variablen, bei denen entschieden werden muß
 - Aufeinandertreffen von verschiedenen Definitionen an Dominatorgrenzen
- Algorithmus nicht trivial ...

OptComp

A. Koch

Organisatorisch

Einleitung

SSA
Erzeugen

SSA für
strukturierte
Sprachen

Dominatoren

Rückwandlung
aus
SSA-Form



Allgemeine Lösung

- Cytron et. al. 1991
- Vorgehen: Berechnen von Dominatorgrenzen
- “Gerade nicht mehr” von Knoten X dominierte Knoten
- Hier nicht mehr klar, ob Definitionen aus X noch gelten
- Einfügen von Phi-Knoten nur für die Variablen, bei denen entschieden werden muß
 - Aufeinandertreffen von verschiedenen Definitionen an Dominatorgrenzen
- Algorithmus nicht trivial ...



Allgemeine Lösung

- Cytron et. al. 1991
- Vorgehen: Berechnen von Dominatorgrenzen
- “Gerade nicht mehr” von Knoten X dominierte Knoten
- Hier nicht mehr klar, ob Definitionen aus X noch gelten
- Einfügen von Phi-Knoten nur für die Variablen, bei denen entschieden werden muß
 - Aufeinandertreffen von verschiedenen Definitionen an Dominatorgrenzen
- Algorithmus nicht trivial ...



Allgemeine Lösung

- Cytron et. al. 1991
- Vorgehen: Berechnen von Dominatorgrenzen
- “Gerade nicht mehr” von Knoten X dominierte Knoten
- Hier nicht mehr klar, ob Definitionen aus X noch gelten
- Einfügen von Phi-Knoten nur für die Variablen, bei denen entschieden werden muß
 - Aufeinandertreffen von verschiedenen Definitionen an Dominatorgrenzen
- Algorithmus nicht trivial ...

Sonderfall: Strukturierte Programmiersprachen



OptComp

A. Koch

Organisatorisch

Einleitung

SSA
Erzeugen

SSA für
strukturierte
Sprachen

Dominatoren

Rückwandlung
aus
SSA-Form

- Keine GOTOs
- Nur strukturierte Anweisungen
 - IF
 - CASE
 - WHILE
 - REPEAT
 - FOR

➔ Viel einfacheres und schnelleres Vorgehen möglich

➔ Brandis/Mössenböck 1994

Unser Ansatz für ein Triangle-Subset!

Sonderfall: Strukturierte Programmiersprachen



- Keine GOTOs
- Nur strukturierte Anweisungen
 - IF
 - CASE
 - WHILE
 - REPEAT
 - FOR

➔ Viel einfacheres und schnelleres Vorgehen möglich

➔ Brandis/Mössenböck 1994

Unser Ansatz für ein Triangle-Subset!

OptComp

A. Koch

Organisatorisch

Einleitung

SSA
Erzeugen

SSA für
strukturierte
Sprachen

Dominatoren

Rückwandlung
aus
SSA-Form



- Keine GOTOs
- Nur strukturierte Anweisungen
 - IF
 - CASE
 - WHILE
 - REPEAT
 - FOR

➔ Viel einfacheres und schnelleres Vorgehen möglich

➔ Brandis/Mössenböck 1994

Unser Ansatz für ein Triangle-Subset!

OptComp

A. Koch

Organisatorisch

Einleitung

SSA
Erzeugen

SSA für
strukturierte
Sprachen

Dominatoren

Rückwandlung
aus
SSA-Form



- Keine GOTOs
- Nur strukturierte Anweisungen
 - IF
 - CASE
 - WHILE
 - REPEAT
 - FOR

➔ Viel einfacheres und schnelleres Vorgehen möglich

➔ Brandis/Mössenböck 1994

Unser Ansatz für ein Triangle-Subset!

OptComp

A. Koch

Organisatorisch

Einleitung

SSA
Erzeugen

SSA für
strukturierte
Sprachen

Dominatoren

Rückwandlung
aus
SSA-Form



- Keine GOTOs
- Nur strukturierte Anweisungen
 - IF
 - CASE
 - WHILE
 - REPEAT
 - FOR

➔ Viel einfacheres und schnelleres Vorgehen möglich

➔ Brandis/Mössenböck 1994

Unser Ansatz für ein Triangle-Subset!

OptComp

A. Koch

Organisatorisch

Einleitung

SSA
Erzeugen

SSA für
strukturierte
Sprachen

Dominatoren

Rückwandlung
aus
SSA-Form



- Keine GOTOs
- Nur strukturierte Anweisungen
 - IF
 - CASE
 - WHILE
 - REPEAT
 - FOR

➔ Viel einfacheres und schnelleres Vorgehen möglich

➔ Brandis/Mössenböck 1994

Unser Ansatz für ein Triangle-Subset!

OptComp

A. Koch

Organisatorisch

Einleitung

SSA
Erzeugen

SSA für
strukturierte
Sprachen

Dominatoren

Rückwandlung
aus
SSA-Form

Sonderfall: Strukturierte Programmiersprachen



- Keine GOTOs
- Nur strukturierte Anweisungen
 - IF
 - CASE
 - WHILE
 - REPEAT
 - FOR

➔ Viel einfacheres und schnelleres Vorgehen möglich

➔ Brandis/Mössenböck 1994

Unser Ansatz für ein Triangle-Subset!

OptComp

A. Koch

Organisatorisch

Einleitung

SSA
Erzeugen

SSA für
strukturierte
Sprachen

Dominatoren

Rückwandlung
aus
SSA-Form

Sonderfall: Strukturierte Programmiersprachen



- Keine GOTOs
- Nur strukturierte Anweisungen
 - IF
 - CASE
 - WHILE
 - REPEAT
 - FOR

➔ Viel einfacheres und schnelleres Vorgehen möglich

➔ Brandis/Mössenböck 1994

Unser Ansatz für ein Triangle-Subset!

OptComp

A. Koch

Organisatorisch

Einleitung

SSA
Erzeugen

SSA für
strukturierte
Sprachen

Dominatoren

Rückwandlung
aus
SSA-Form



- Keine GOTOs
- Nur strukturierte Anweisungen
 - IF
 - CASE
 - WHILE
 - REPEAT
 - FOR

➔ Viel einfacheres und schnelleres Vorgehen möglich

➔ Brandis/Mössenböck 1994

Unser Ansatz für ein Triangle-Subset!

OptComp

A. Koch

Organisatorisch

Einleitung

SSA
Erzeugen

SSA für
strukturierte
Sprachen

Dominatoren

Rückwandlung
aus
SSA-Form



- Keine GOTOs
- Nur strukturierte Anweisungen
 - IF
 - CASE
 - WHILE
 - REPEAT
 - FOR

↳ Viel einfacheres und schnelleres Vorgehen möglich

↳ Brandis/Mössenböck 1994

Unser Ansatz für ein Triangle-Subset!



- Keine GOTOs
- Nur strukturierte Anweisungen
 - IF
 - CASE
 - WHILE
 - REPEAT
 - FOR

➔ Viel einfacheres und schnelleres Vorgehen möglich

➔ Brandis/Mössenböck 1994

Unser Ansatz für ein Triangle-Subset!



Aus Zeitgründen in OC06 **keine** Behandlung von

- Arrays
- Records
- Prozeduraufrufen
- Verschachtelten Geltungsbereichen

Alles handhabbar!

... aber aufwändig und lenkt von Kernideen ab.

Bei Interesse: Cytron et al., Abschnitt 3.1



Aus Zeitgründen in OC06 **keine** Behandlung von

- Arrays
- Records
- Prozeduraufrufen
- Verschachtelten Geltungsbereichen

Alles handhabbar!

... aber aufwändig und lenkt von Kernideen ab.

Bei Interesse: Cytron et al., Abschnitt 3.1



Aus Zeitgründen in OC06 **keine** Behandlung von

- Arrays
- Records
- Prozeduraufrufen
- Verschachtelten Geltungsbereichen

Alles handhabbar!

... aber aufwändig und lenkt von Kernideen ab.

Bei Interesse: Cytron et al., Abschnitt 3.1



Aus Zeitgründen in OC06 **keine** Behandlung von

- Arrays
- Records
- Prozeduraufrufen
- Verschachtelten Geltungsbereichen

Alles handhabbar!

... aber aufwändig und lenkt von Kernideen ab.

Bei Interesse: Cytron et al., Abschnitt 3.1



Aus Zeitgründen in OC06 **keine** Behandlung von

- Arrays
- Records
- Prozeduraufrufen
- Verschachtelten Geltungsbereichen

Alles handhabbar!

... aber aufwändig und lenkt von Kernideen ab.

Bei Interesse: Cytron et al., Abschnitt 3.1



Aus Zeitgründen in OC06 **keine** Behandlung von

- Arrays
- Records
- Prozeduraufrufen
- Verschachtelten Geltungsbereichen

Alles handhabbar!

... aber aufwändig und lenkt von Kernideen ab.

Bei Interesse: Cytron et al., Abschnitt 3.1



Aus Zeitgründen in OC06 **keine** Behandlung von

- Arrays
- Records
- Prozeduraufrufen
- Verschachtelten Geltungsbereichen

Alles handhabbar!

... aber aufwändig und lenkt von Kernideen ab.

Bei Interesse: Cytron et al., Abschnitt 3.1



Aus Zeitgründen in OC06 **keine** Behandlung von

- Arrays
- Records
- Prozeduraufrufen
- Verschachtelten Geltungsbereichen

Alles handhabbar!

... aber aufwändig und lenkt von Kernideen ab.

Bei Interesse: Cytron et al., Abschnitt 3.1

Benennen von Werten in Basisblöcken



<i>Assignments (original form)</i>	<i>Assignments (SSA form)</i>	<i>Current values</i>	
		<i>v</i>	<i>x</i>
		v_0	x_0
$v := 0;$	$v_1 := 0;$	v_1	x_0
$x := v + 1;$	$x_1 := v_1 + 1;$	v_1	x_1
$v := 2$	$v_2 := 2$	v_2	x_1

- Jede Zuweisung an v erzeugt neuen Wert v_i
- Nach Zuweisung ist v_i aktueller Wert von v
- Ersetze alle folgenden Verwendungen von v durch v_i
- Verwaltung z.B. in extra Tabelle während Umformung

Benennen von Werten in Basisblöcken



<i>Assignments (original form)</i>	<i>Assignments (SSA form)</i>	<i>Current values</i>	
		<i>v</i>	<i>x</i>
		v_0	x_0
$v := 0;$	$v_1 := 0;$	v_1	x_0
$x := v + 1;$	$x_1 := v_1 + 1;$	v_1	x_1
$v := 2$	$v_2 := 2$	v_2	x_1

- Jede Zuweisung an v erzeugt neuen Wert v_i
- Nach Zuweisung ist v_i **aktueller** Wert von v
- Ersetze alle **folgenden** Verwendungen von v durch v_i
- Verwaltung z.B. in extra Tabelle während Umformung

Benennen von Werten in Basisblöcken



<i>Assignments (original form)</i>	<i>Assignments (SSA form)</i>	<i>Current values</i>	
		<i>v</i>	<i>x</i>
$v := 0;$	$v_1 := 0;$	v_0	x_0
$x := v + 1;$	$x_1 := v_1 + 1;$	v_1	x_0
$v := 2$	$v_2 := 2$	v_1	x_1
		v_2	x_1

- Jede Zuweisung an v erzeugt neuen Wert v_i
- Nach Zuweisung ist v_i **aktueller** Wert von v
- Ersetze alle **folgenden** Verwendungen von v durch v_i
- Verwaltung z.B. in extra Tabelle während Umformung

Benennen von Werten in Basisblöcken



<i>Assignments (original form)</i>	<i>Assignments (SSA form)</i>	<i>Current values</i>	
		<i>v</i>	<i>x</i>
		v_0	x_0
$v := 0;$	$v_1 := 0;$	v_1	x_0
$x := v + 1;$	$x_1 := v_1 + 1;$	v_1	x_1
$v := 2$	$v_2 := 2$	v_2	x_1

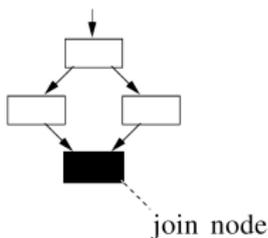
- Jede Zuweisung an v erzeugt neuen Wert v_i
- Nach Zuweisung ist v_i **aktueller** Wert von v
- Ersetze alle **folgenden** Verwendungen von v durch v_i
- Verwaltung z.B. in extra Tabelle während Umformung



Bei strukturierten Programmiersprachen:

Alle Join-Knoten sind durch Konstrukte bereits vorgegeben

IF, CASE



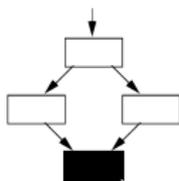
Platzhalter
(siehe später!)



Bei strukturierten Programmiersprachen:

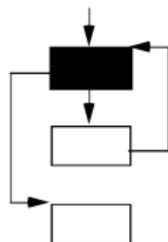
Alle Join-Knoten sind durch Konstrukte bereits vorgegeben

IF, CASE



join node

WHILE, FOR



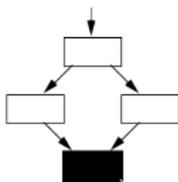
Platzhalter
(siehe später!)



Bei strukturierten Programmiersprachen:

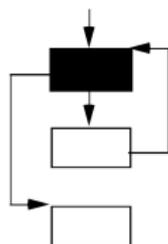
Alle Join-Knoten sind durch Konstrukte bereits vorgegeben

IF, CASE

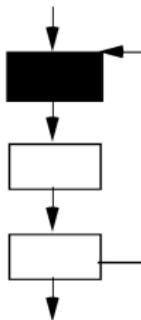


join node

WHILE, FOR



REPEAT



Platzhalter
(siehe später!)

Join-Knoten 1



OptComp

A. Koch

Organisatorisch

Einleitung

SSA

Erzeugen

SSA für
strukturierte
Sprachen

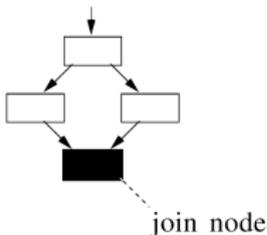
Dominatoren

Rückwandlung
aus
SSA-Form

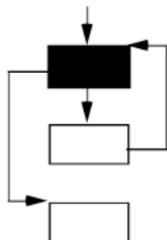
Bei strukturierten Programmiersprachen:

Alle Join-Knoten sind durch Konstrukte bereits vorgegeben

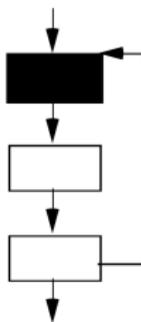
IF, CASE



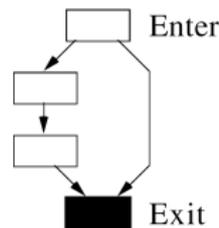
WHILE, FOR



REPEAT



procedure



Platzhalter
(siehe später!)

Join-Knoten 1



OptComp

A. Koch

Organisatorisch

Einleitung

SSA

Erzeugen

SSA für
strukturierte
Sprachen

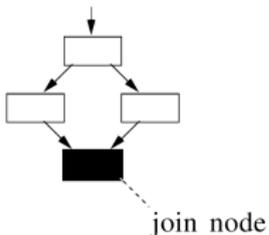
Dominatoren

Rückwandlung
aus
SSA-Form

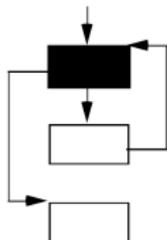
Bei strukturierten Programmiersprachen:

Alle Join-Knoten sind durch Konstrukte bereits vorgegeben

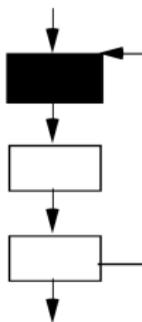
IF, CASE



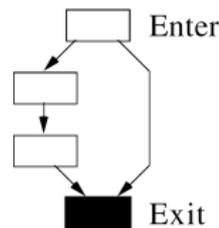
WHILE, FOR



REPEAT



procedure



Platzhalter
(siehe später!)

Join-Knoten 1



OptComp

A. Koch

Organisatorisch

Einleitung

SSA

Erzeugen

SSA für
strukturierte
Sprachen

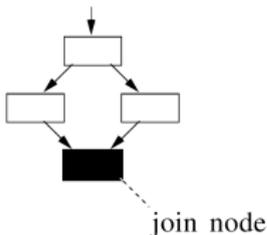
Dominatoren

Rückwandlung
aus
SSA-Form

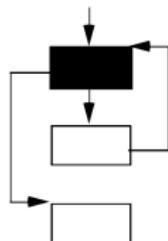
Bei strukturierten Programmiersprachen:

Alle Join-Knoten sind durch Konstrukte bereits vorgegeben

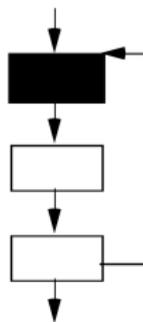
IF, CASE



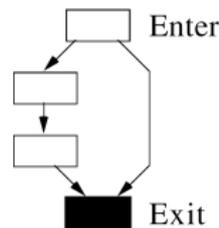
WHILE, FOR



REPEAT



procedure



Platzhalter
(siehe später!)



- **Strukturen können verschachtelt sein**
 - Bearbeite von **innen** nach **aussen**
 - Innerster Join-Knoten ist **aktueller** Join-Knoten
- Erzeuge keine **speziellen** Knoten für Joins
- Verwende bisherige Blöcke weiter

OptComp

A. Koch

Organisatorisch

Einleitung

SSA

Erzeugen

SSA für
strukturierte
Sprachen

Dominatoren

Rückwandlung
aus
SSA-Form



OptComp

A. Koch

Organisatorisch

Einleitung

SSA
Erzeugen

SSA für
strukturierte
Sprachen

Dominatoren

Rückwandlung
aus
SSA-Form

- Strukturen können verschachtelt sein
 - Bearbeite von **innen** nach **aussen**
 - Innerster Join-Knoten ist **aktueller** Join-Knoten
- Erzeuge keine **speziellen** Knoten für Joins
- Verwende bisherige Blöcke weiter



- Strukturen können verschachtelt sein
 - Bearbeite von **innen** nach **aussen**
 - Innerster Join-Knoten ist **aktueller** Join-Knoten
- Erzeuge keine **speziellen** Knoten für Joins
- Verwende bisherige Blöcke weiter



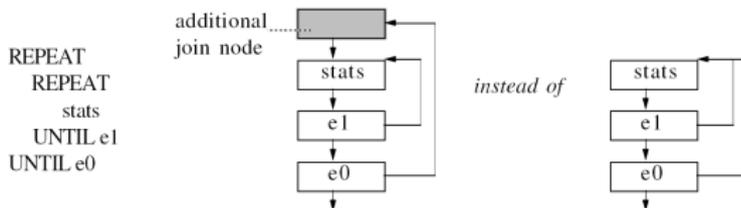
- Strukturen können verschachtelt sein
 - Bearbeite von **innen** nach **aussen**
 - Innerster Join-Knoten ist **aktueller** Join-Knoten
- Erzeuge keine **speziellen** Knoten für Joins
- Verwende bisherige Blöcke weiter



- Strukturen können verschachtelt sein
 - Bearbeite von **innen** nach **aussen**
 - Innerster Join-Knoten ist **aktueller** Join-Knoten
- Erzeuge keine **speziellen** Knoten für Joins
- Verwende bisherige Blöcke weiter



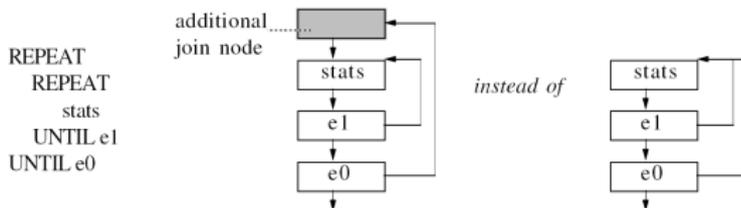
Ausnahme: Verschachtelte REPEAT-Anweisungen



- Für spätere Optimierung hilfreich
- Sonst kein Ziel für aus der inneren Schleife bewegte Berechnungen



Ausnahme: Verschachtelte REPEAT-Anweisungen



- Für spätere Optimierung hilfreich
- Sonst kein Ziel für aus der inneren Schleife bewegte Berechnungen

Phi-Knoten Erzeugen 1



OptComp

A. Koch

Organisatorisch

Einleitung

SSA
Erzeugen

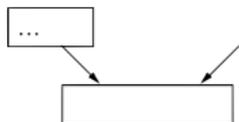
SSA für
strukturierte
Sprachen

Dominatoren

Rückwandlung
aus
SSA-Form

- **Jede** Zuweisung gehört zu einem Zweig des Kontrollflußgraphen
- Jede Zuweisung erzeugt einen neuen Wert
- Irgendwann trifft der Wert auf einen Join-Knoten
- Hier Unterscheidung zwischen allen Werten für diese Variable

➔ Jede Zuweisung erzeugt oder modifiziert Phi-Funktion für Variable



Phi-Knoten Erzeugen 1



OptComp

A. Koch

Organisatorisch

Einleitung

SSA

Erzeugen

SSA für
strukturierte
Sprachen

Dominatoren

Rückwandlung
aus
SSA-Form

- Jede Zuweisung gehört zu einem Zweig des Kontrollflußgraphen
- Jede Zuweisung erzeugt einen neuen Wert
- Irgendwann trifft der Wert auf einen Join-Knoten
- Hier Unterscheidung zwischen allen Werten für diese Variable

➔ Jede Zuweisung erzeugt oder modifiziert Phi-Funktion für Variable



Phi-Knoten Erzeugen 1



OptComp

A. Koch

Organisatorisch

Einleitung

SSA

Erzeugen

SSA für
strukturierte
Sprachen

Dominatoren

Rückwandlung
aus
SSA-Form

- Jede Zuweisung gehört zu einem Zweig des Kontrollflußgraphen
- Jede Zuweisung erzeugt einen neuen Wert
- Irgendwann trifft der Wert auf einen Join-Knoten
- Hier Unterscheidung zwischen allen Werten für diese Variable

➔ Jede Zuweisung erzeugt oder modifiziert Phi-Funktion für Variable



Phi-Knoten Erzeugen 1



OptComp

A. Koch

Organisatorisch

Einleitung

SSA

Erzeugen

SSA für
strukturierte
Sprachen

Dominatoren

Rückwandlung
aus
SSA-Form

- **Jede** Zuweisung gehört zu einem Zweig des Kontrollflußgraphen
- Jede Zuweisung erzeugt einen neuen Wert
- Irgendwann trifft der Wert auf einen Join-Knoten
- Hier Unterscheidung zwischen allen Werten für diese Variable

➔ Jede Zuweisung erzeugt oder modifiziert Phi-Funktion für Variable



Phi-Knoten Erzeugen 1



OptComp

A. Koch

Organisatorisch

Einleitung

SSA

Erzeugen

SSA für
strukturierte
Sprachen

Dominatoren

Rückwandlung
aus
SSA-Form

- Jede Zuweisung gehört zu einem Zweig des Kontrollflußgraphen
- Jede Zuweisung erzeugt einen neuen Wert
- Irgendwann trifft der Wert auf einen Join-Knoten
- Hier Unterscheidung zwischen allen Werten für diese Variable

➔ Jede Zuweisung erzeugt oder modifiziert Phi-Funktion für Variable

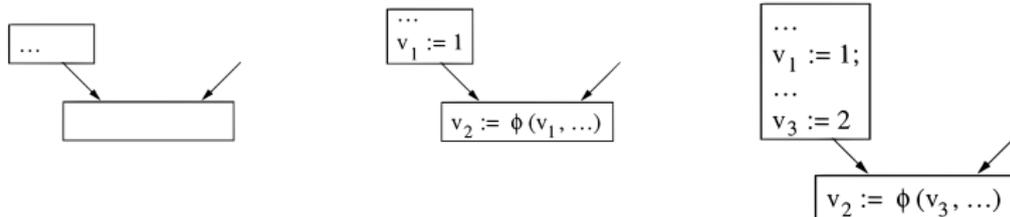


Phi-Knoten Erzeugen 1



- Jede Zuweisung gehört zu einem Zweig des Kontrollflußgraphen
- Jede Zuweisung erzeugt einen neuen Wert
- Irgendwann trifft der Wert auf einen Join-Knoten
- Hier Unterscheidung zwischen allen Werten für diese Variable

➔ Jede Zuweisung erzeugt oder modifiziert Phi-Funktion für Variable



Phi-Knoten Erzeugen 2



- Phi-Operand entsprechend dem bearbeiteten Zweig
- ... wird jeweils auf letzten aktuellen Wert gesetzt
- Phi-Funktionen treten selber in Zuweisungen auf
- Erzeugen also selber neue Werte
- Führen zu weiteren Phi-Funktionen in nächstäußerem Join-Knoten
- Ende bei Erreichen des Exit-Knotens

Vorgehen: Erzeugen des CFGs in SSA-Form durch Traversieren des ASTs

Könnte aber auch direkt beim Parsen geschehen.

OptComp

A. Koch

Organisatorisch

Einleitung

SSA

Erzeugen

SSA für
strukturierte
Sprachen

Dominatoren

Rückwandlung
aus
SSA-Form

Phi-Knoten Erzeugen 2



- Phi-Operand entsprechend dem bearbeiteten Zweig
- ... wird jeweils auf letzten aktuellen Wert gesetzt
- Phi-Funktionen treten selber in Zuweisungen auf
- Erzeugen also selber neue Werte
- Führen zu weiteren Phi-Funktionen in nächstäußerem Join-Knoten
- Ende bei Erreichen des Exit-Knotens

Vorgehen: Erzeugen des CFGs in SSA-Form durch Traversieren des ASTs

Könnte aber auch direkt beim Parsen geschehen.

OptComp

A. Koch

Organisatorisch

Einleitung

SSA

Erzeugen

SSA für
strukturierte
Sprachen

Dominatoren

Rückwandlung
aus
SSA-Form

Phi-Knoten Erzeugen 2



- Phi-Operand entsprechend dem bearbeiteten Zweig
- ... wird jeweils auf letzten aktuellen Wert gesetzt
- Phi-Funktionen treten selber in Zuweisungen auf
- Erzeugen also selber neue Werte
- Führen zu weiteren Phi-Funktionen in nächstäußerem Join-Knoten
- Ende bei Erreichen des Exit-Knotens

Vorgehen: Erzeugen des CFGs in SSA-Form durch Traversieren des ASTs

Könnte aber auch direkt beim Parsen geschehen.

OptComp

A. Koch

Organisatorisch

Einleitung

SSA

Erzeugen

SSA für
strukturierte
Sprachen

Dominatoren

Rückwandlung
aus
SSA-Form

Phi-Knoten Erzeugen 2



- Phi-Operand entsprechend dem bearbeiteten Zweig
- ... wird jeweils auf letzten aktuellen Wert gesetzt
- Phi-Funktionen treten selber in Zuweisungen auf
- Erzeugen also selber neue Werte
- Führen zu weiteren Phi-Funktionen in nächstäußerem Join-Knoten
- Ende bei Erreichen des Exit-Knotens

Vorgehen: Erzeugen des CFGs in SSA-Form durch Traversieren des ASTs

Könnte aber auch direkt beim Parsen geschehen.

OptComp

A. Koch

Organisatorisch

Einleitung

SSA

Erzeugen

SSA für
strukturierte
Sprachen

Dominatoren

Rückwandlung
aus
SSA-Form

Phi-Knoten Erzeugen 2



- Phi-Operand entsprechend dem bearbeiteten Zweig
- ... wird jeweils auf letzten aktuellen Wert gesetzt
- Phi-Funktionen treten selber in Zuweisungen auf
- Erzeugen also selber neue Werte
- Führen zu weiteren Phi-Funktionen in nächstäußerem Join-Knoten
- Ende bei Erreichen des Exit-Knotens

Vorgehen: Erzeugen des CFGs in SSA-Form durch Traversieren des ASTs

Könnte aber auch direkt beim Parsen geschehen.

OptComp

A. Koch

Organisatorisch

Einleitung

SSA

Erzeugen

SSA für
strukturierte
Sprachen

Dominatoren

Rückwandlung
aus
SSA-Form

Phi-Knoten Erzeugen 2



- Phi-Operand entsprechend dem bearbeiteten Zweig
- ... wird jeweils auf letzten aktuellen Wert gesetzt
- Phi-Funktionen treten selber in Zuweisungen auf
- Erzeugen also selber neue Werte
- Führen zu weiteren Phi-Funktionen in nächstäußerem Join-Knoten
- Ende bei Erreichen des Exit-Knotens

Vorgehen: Erzeugen des CFGs in SSA-Form durch Traversieren des ASTs

Könnte aber auch direkt beim Parsen geschehen.

OptComp

A. Koch

Organisatorisch

Einleitung

SSA

Erzeugen

SSA für
strukturierte
Sprachen

Dominatoren

Rückwandlung
aus
SSA-Form

Phi-Knoten Erzeugen 2



- Phi-Operand entsprechend dem bearbeiteten Zweig
- ... wird jeweils auf letzten aktuellen Wert gesetzt
- Phi-Funktionen treten selber in Zuweisungen auf
- Erzeugen also selber neue Werte
- Führen zu weiteren Phi-Funktionen in nächstäußerem Join-Knoten
- Ende bei Erreichen des Exit-Knotens

Vorgehen: Erzeugen des CFGs in SSA-Form durch Traversieren des ASTs

Könnte aber auch direkt beim Parsen geschehen.

OptComp

A. Koch

Organisatorisch

Einleitung

SSA

Erzeugen

SSA für
strukturierte
Sprachen

Dominatoren

Rückwandlung
aus
SSA-Form

Phi-Knoten Erzeugen 2



- Phi-Operand entsprechend dem bearbeiteten Zweig
- ... wird jeweils auf letzten aktuellen Wert gesetzt
- Phi-Funktionen treten selber in Zuweisungen auf
- Erzeugen also selber neue Werte
- Führen zu weiteren Phi-Funktionen in nächstäußerem Join-Knoten
- Ende bei Erreichen des Exit-Knotens

Vorgehen: Erzeugen des CFGs in SSA-Form durch Traversieren des ASTs

Könnte aber auch direkt beim Parsen geschehen.

OptComp

A. Koch

Organisatorisch

Einleitung

SSA

Erzeugen

SSA für
strukturierte
Sprachen

Dominatoren

Rückwandlung
aus
SSA-Form

Phi-Knoten Erzeugen 2



- Phi-Operand entsprechend dem bearbeiteten Zweig
- ... wird jeweils auf letzten aktuellen Wert gesetzt
- Phi-Funktionen treten selber in Zuweisungen auf
- Erzeugen also selber neue Werte
- Führen zu weiteren Phi-Funktionen in nächstäußerem Join-Knoten
- Ende bei Erreichen des Exit-Knotens

Vorgehen: Erzeugen des CFGs in SSA-Form durch Traversieren des ASTs

Könnte aber auch direkt beim Parsen geschehen.

OptComp

A. Koch

Organisatorisch

Einleitung

SSA

Erzeugen

SSA für
strukturierte
Sprachen

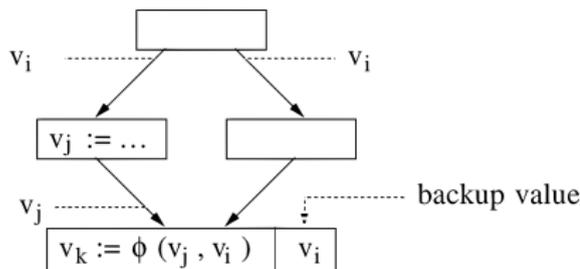
Dominatoren

Rückwandlung
aus
SSA-Form

Phi-Knoten für IF-Anweisungen 1



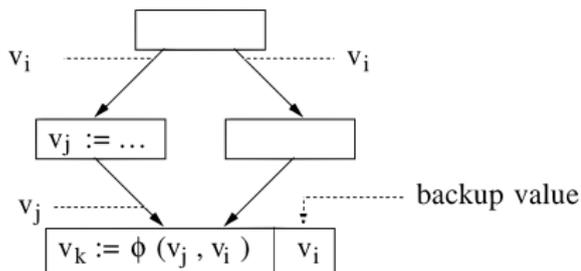
- 1 Bei Erreichen von IF: Erzeuge neuen Join-Knoten
 - Wird Phi-Funktionen aus THEN/ELSE enthalten
 - Wird später in den CFG eingehängt
- 2 Bearbeite THEN-Zweig, für eine Zuweisung an v
 - 1. Mal: Lege leere Phi-Funktion (Identität) für v an, sichere Wert v_i vor IF zusammen mit Phi-Funktion
 - Sonst: Setze Phi-Operand auf jeweils aktuellen Wert v_j
- 3 Bearbeite ELSE-Zweig
 - Setze aktuelle auf gesicherte Werte (pre-IF) zurück
 - Dann gleiches Vorgehen wie im THEN-Zweig



Phi-Knoten für IF-Anweisungen 1



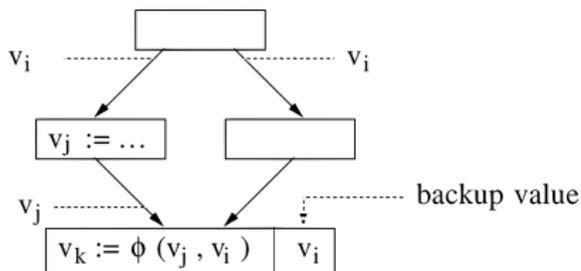
- 1 Bei Erreichen von IF: Erzeuge neuen Join-Knoten
 - Wird Phi-Funktionen aus THEN/ELSE enthalten
 - Wird später in den CFG eingehängt
- 2 Bearbeite THEN-Zweig, für eine Zuweisung an v
 - 1. Mal: Lege leere Phi-Funktion (Identität) für v an, sichere Wert v_i vor IF zusammen mit Phi-Funktion
 - Sonst: Setze Phi-Operand auf jeweils aktuellen Wert v_j
- 3 Bearbeite ELSE-Zweig
 - Setze aktuelle auf gesicherte Werte (pre-IF) zurück
 - Dann gleiches Vorgehen wie im THEN-Zweig



Phi-Knoten für IF-Anweisungen 1



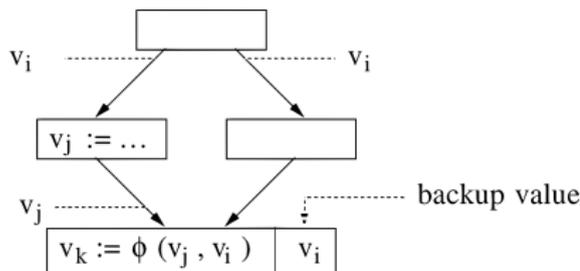
- 1 Bei Erreichen von IF: Erzeuge neuen Join-Knoten
 - Wird Phi-Funktionen aus THEN/ELSE enthalten
 - Wird später in den CFG eingehängt
- 2 Bearbeite THEN-Zweig, für eine Zuweisung an v
 - 1. Mal: Lege leere Phi-Funktion (Identität) für v an, sichere Wert v_i vor IF zusammen mit Phi-Funktion
 - Sonst: Setze Phi-Operand auf jeweils aktuellen Wert v_j
- 3 Bearbeite ELSE-Zweig
 - Setze aktuelle auf gesicherte Werte (pre-IF) zurück
 - Dann gleiches Vorgehen wie im THEN-Zweig



Phi-Knoten für IF-Anweisungen 1



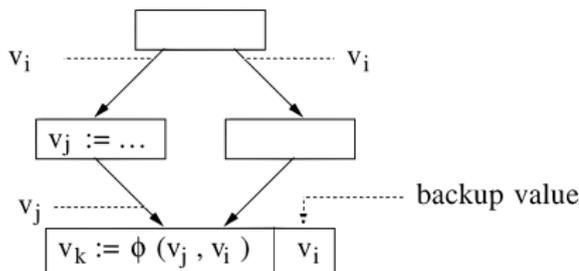
- 1 Bei Erreichen von IF: Erzeuge neuen Join-Knoten
 - Wird Phi-Funktionen aus THEN/ELSE enthalten
 - Wird später in den CFG eingehängt
- 2 Bearbeite THEN-Zweig, für eine Zuweisung an v
 - 1. Mal: Lege leere Phi-Funktion (Identität) für v an, sichere Wert v_j vor IF zusammen mit Phi-Funktion
 - Sonst: Setze Phi-Operand auf jeweils aktuellen Wert v_j
- 3 Bearbeite ELSE-Zweig
 - Setze aktuelle auf gesicherte Werte (pre-IF) zurück
 - Dann gleiches Vorgehen wie im THEN-Zweig



Phi-Knoten für IF-Anweisungen 1



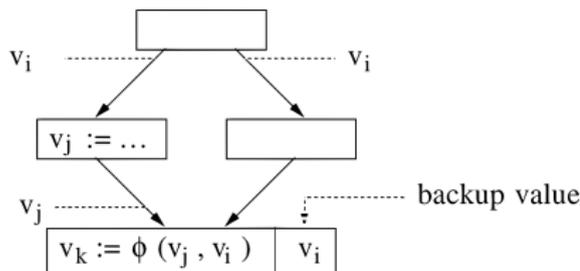
- 1 Bei Erreichen von IF: Erzeuge neuen Join-Knoten
 - Wird Phi-Funktionen aus THEN/ELSE enthalten
 - Wird später in den CFG eingehängt
- 2 Bearbeite THEN-Zweig, für eine Zuweisung an v
 - 1. Mal: Lege leere Phi-Funktion (Identität) für v an, sichere Wert v_i vor IF zusammen mit Phi-Funktion
 - Sonst: Setze Phi-Operand auf jeweils aktuellen Wert v_j
- 3 Bearbeite ELSE-Zweig
 - Setze aktuelle auf gesicherte Werte (pre-IF) zurück
 - Dann gleiches Vorgehen wie im THEN-Zweig



Phi-Knoten für IF-Anweisungen 1



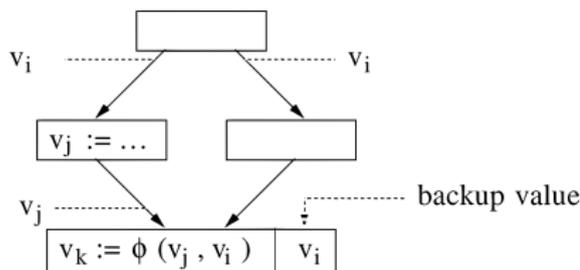
- 1 Bei Erreichen von IF: Erzeuge neuen Join-Knoten
 - Wird Phi-Funktionen aus THEN/ELSE enthalten
 - Wird später in den CFG eingehängt
- 2 Bearbeite THEN-Zweig, für eine Zuweisung an v
 - 1. Mal: Lege leere Phi-Funktion (Identität) für v an, sichere Wert v_i vor IF zusammen mit Phi-Funktion
 - Sonst: Setze Phi-Operand auf jeweils aktuellen Wert v_j
- 3 Bearbeite ELSE-Zweig
 - Setze aktuelle auf gesicherte Werte (pre-IF) zurück
 - Dann gleiches Vorgehen wie im THEN-Zweig



Phi-Knoten für IF-Anweisungen 1



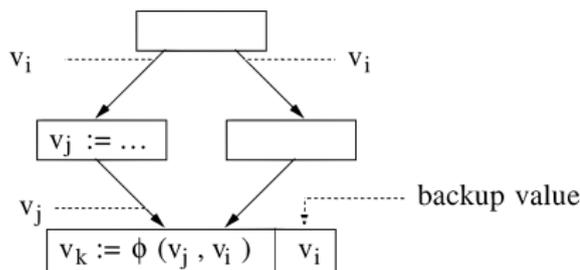
- 1 Bei Erreichen von IF: Erzeuge neuen Join-Knoten
 - Wird Phi-Funktionen aus THEN/ELSE enthalten
 - Wird später in den CFG eingehängt
- 2 Bearbeite THEN-Zweig, für eine Zuweisung an v
 - 1. Mal: Lege leere Phi-Funktion (Identität) für v an, sichere Wert v_i vor IF zusammen mit Phi-Funktion
 - Sonst: Setze Phi-Operand auf jeweils aktuellen Wert v_j
- 3 Bearbeite ELSE-Zweig
 - Setze aktuelle auf gesicherte Werte (pre-IF) zurück
 - Dann gleiches Vorgehen wie im THEN-Zweig



Phi-Knoten für IF-Anweisungen



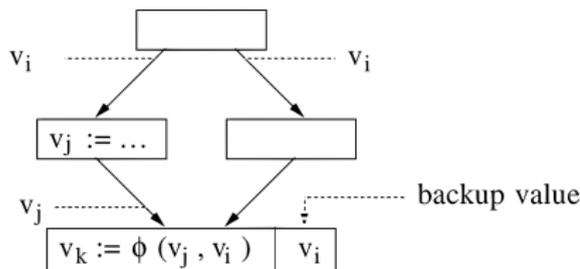
- 1 Bei Erreichen von IF: Erzeuge neuen Join-Knoten
 - Wird Phi-Funktionen aus THEN/ELSE enthalten
 - Wird später in den CFG eingehängt
- 2 Bearbeite THEN-Zweig, für eine Zuweisung an v
 - 1. Mal: Lege leere Phi-Funktion (Identität) für v an, sichere Wert v_i vor IF zusammen mit Phi-Funktion
 - Sonst: Setze Phi-Operand auf jeweils aktuellen Wert v_j
- 3 Bearbeite ELSE-Zweig
 - Setze aktuelle auf gesicherte Werte (pre-IF) zurück
 - Dann gleiches Vorgehen wie im THEN-Zweig



Phi-Knoten für IF-Anweisungen 1



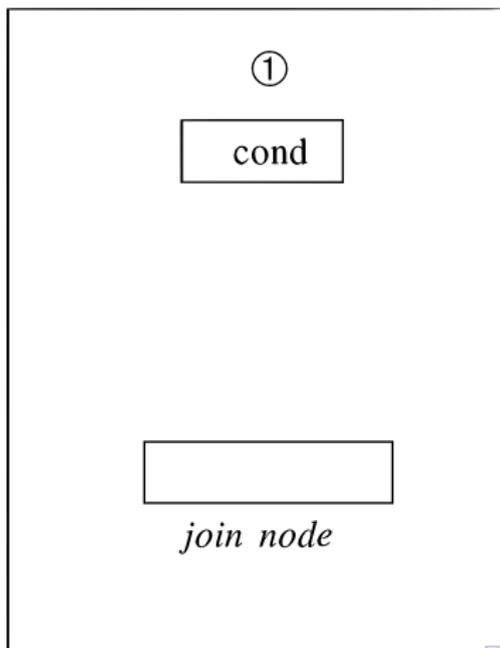
- 1 Bei Erreichen von IF: Erzeuge neuen Join-Knoten
 - Wird Phi-Funktionen aus THEN/ELSE enthalten
 - Wird später in den CFG eingehängt
- 2 Bearbeite THEN-Zweig, für eine Zuweisung an v
 - 1. Mal: Lege leere Phi-Funktion (Identität) für v an, sichere Wert v_i vor IF zusammen mit Phi-Funktion
 - Sonst: Setze Phi-Operand auf jeweils aktuellen Wert v_j
- 3 Bearbeite ELSE-Zweig
 - Setze aktuelle auf gesicherte Werte (pre-IF) zurück
 - Dann gleiches Vorgehen wie im THEN-Zweig



Phi-Knoten für IF-Anweisungen 2



IF cond THEN a := 1; b := a + 1 ELSE a := a + 1; c := 2 END



OptComp

A. Koch

Organisatorisch

Einleitung

SSA
Erzeugen

SSA für
strukturierte
Sprachen

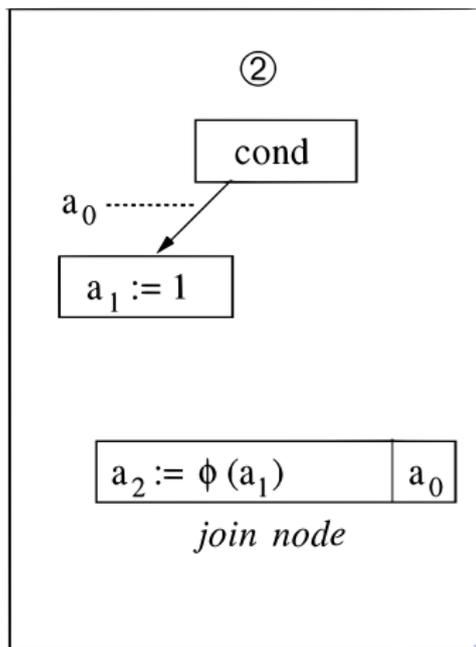
Dominatoren

Rückwandlung
aus
SSA-Form

Phi-Knoten für IF-Anweisungen 2



①
IF cond THEN a := 1; b := a + 1
②
③ ELSE a := a + 1; c := 2
④
⑤ END



OptComp

A. Koch

Organisatorisch

Einleitung

SSA
Erzeugen

SSA für
strukturierte
Sprachen

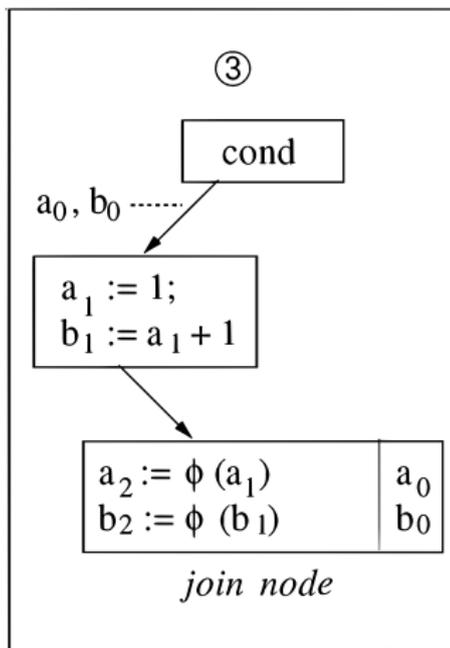
Dominatoren

Rückwandlung
aus
SSA-Form

Phi-Knoten für IF-Anweisungen 2



①
IF cond THEN a := 1; b := a + 1
②
ELSE a := a + 1; c := 2
③
END
④
⑤



OptComp

A. Koch

Organisatorisch

Einleitung

SSA
Erzeugen

SSA für
strukturierte
Sprachen

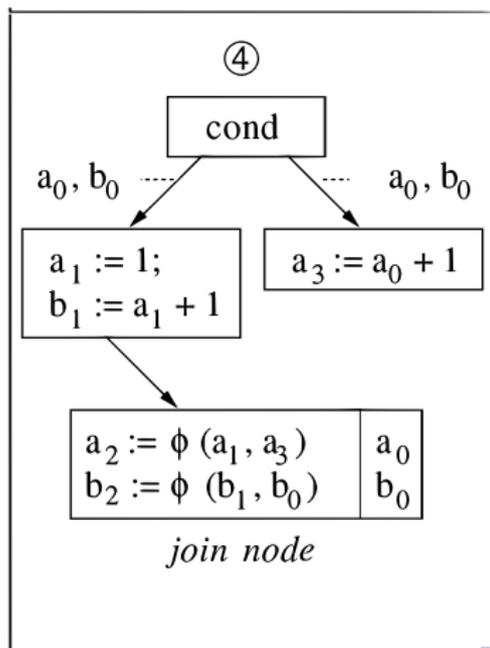
Dominatoren

Rückwandlung
aus
SSA-Form

Phi-Knoten für IF-Anweisungen 2



① IF cond THEN a := 1; b := a + 1 ELSE a := a + 1; c := 2 END
②
③
④
⑤



OptComp

A. Koch

Organisatorisch

Einleitung

SSA
Erzeugen

SSA für
strukturierte
Sprachen

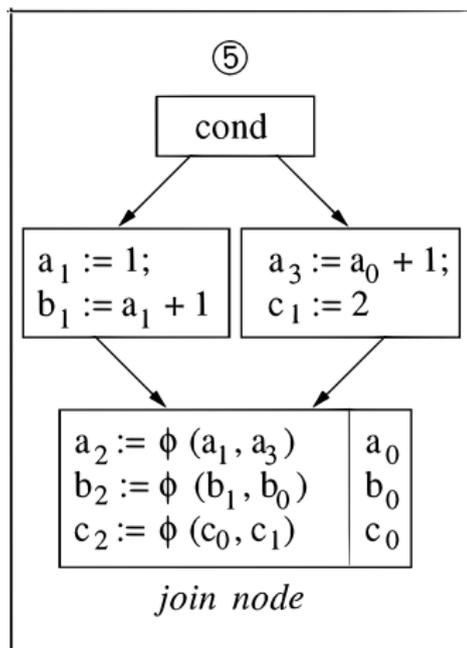
Dominatoren

Rückwandlung
aus
SSA-Form

Phi-Knoten für IF-Anweisungen 2



① ② ③ ④ ⑤
IF cond THEN a := 1; b := a + 1 ELSE a := a + 1; c := 2 END



OptComp

A. Koch

Organisatorisch

Einleitung

SSA
Erzeugen

SSA für
strukturierte
Sprachen

Dominatoren

Rückwandlung
aus
SSA-Form



Nach Abarbeiten von THEN und ELSE-Zweigen:

Festlegen des Join-Blocks (*commit*)

- Join-Block **selber** bearbeiten
- Werte Zuweisungen von Phi-Funktionen aus
- Trage neue Phi-Funktionen in **nächstäußeren** Join-Block ein
- Trage Join-Block des umschließenden Kontrollstruktur
- Trage LHS der Phi-Zuweisungen als aktuelle Werte der Variablen ein
- Hänge aktuellen Join-Block in CFG ein



Nach Abarbeiten von THEN und ELSE-Zweigen:
Festlegen des Join-Blocks (*commit*)

- Join-Block **selber** bearbeiten
- Werte Zuweisungen von Phi-Funktionen aus
- Trage neue Phi-Funktionen in **nächstäußeren** Join-Block ein
 - Join-Block der umschließenden Kontrollstruktur
- Trage LHS der Phi-Zuweisungen als aktuelle Werte der Variablen ein
- Hänge aktuellen Join-Block in CFG ein



Nach Abarbeiten von THEN und ELSE-Zweigen:
Festlegen des Join-Blocks (*commit*)

- Join-Block **selber** bearbeiten
- Werte Zuweisungen von Phi-Funktionen aus
- Trage neue Phi-Funktionen in **nächstäußeren** Join-Block ein
 - Join-Block der umschließenden Kontrollstruktur
- Trage LHS der Phi-Zuweisungen als aktuelle Werte der Variablen ein
- Hänge aktuellen Join-Block in CFG ein



Nach Abarbeiten von THEN und ELSE-Zweigen:
Festlegen des Join-Blocks (*commit*)

- Join-Block **selber** bearbeiten
- Werte Zuweisungen von Phi-Funktionen aus
- Trage neue Phi-Funktionen in **nächstäußeren** Join-Block ein
 - Join-Block der umschließenden Kontrollstruktur
- Trage LHS der Phi-Zuweisungen als aktuelle Werte der Variablen ein
- Hänge aktuellen Join-Block in CFG ein



Nach Abarbeiten von THEN und ELSE-Zweigen:
Festlegen des Join-Blocks (*commit*)

- Join-Block **selber** bearbeiten
- Werte Zuweisungen von Phi-Funktionen aus
- Trage neue Phi-Funktionen in **nächstäußeren** Join-Block ein
 - Join-Block der umschließenden Kontrollstruktur
- Trage LHS der Phi-Zuweisungen als aktuelle Werte der Variablen ein
- Hänge aktuellen Join-Block in CFG ein



Nach Abarbeiten von THEN und ELSE-Zweigen:
Festlegen des Join-Blocks (*commit*)

- Join-Block **selber** bearbeiten
- Werte Zuweisungen von Phi-Funktionen aus
- Trage neue Phi-Funktionen in **nächstäußeren** Join-Block ein
 - Join-Block der umschließenden Kontrollstruktur
- Trage LHS der Phi-Zuweisungen als aktuelle Werte der Variablen ein
- Hänge aktuellen Join-Block in CFG ein



Nach Abarbeiten von THEN und ELSE-Zweigen:
Festlegen des Join-Blocks (*commit*)

- Join-Block **selber** bearbeiten
- Werte Zuweisungen von Phi-Funktionen aus
- Trage neue Phi-Funktionen in **nächstäußeren** Join-Block ein
 - Join-Block der umschließenden Kontrollstruktur
- Trage LHS der Phi-Zuweisungen als aktuelle Werte der Variablen ein
- Hänge aktuellen Join-Block in CFG ein

Phi-Knoten in WHILE-Anweisungen 1



- Join-Knoten von WHILE-Anweisung ist **Kopfknoten**
 - Zusammentreffen von Schleifeneintritt und Rückwärtskante im CFG
- Bearbeitung des Schleifenkörpers analog zur IF-Anweisung, **aber**
- Bei Eintragen einer neuen Phi-Funktion in Kopfknoten
- ... entsteht **neuer** aktueller Wert
- Alle Benutzungen der Variable im Schleifenkörper durch aktuellen Wert ersetzen
 - Verwalte Liste aller im Schleifenkörper benutzten Werte
 - Sogenannte *use chain*
 - Kann für schnelle Korrektur (ändern der Versionsnummer) benutzt werden

OptComp

A. Koch

Organisatorisch

Einleitung

SSA
Erzeugen

SSA für
strukturierte
Sprachen

Dominatoren

Rückwandlung
aus
SSA-Form

Phi-Knoten in WHILE-Anweisungen 1



- Join-Knoten von WHILE-Anweisung ist **Kopfknoten**
 - Zusammentreffen von Schleifeneintritt und Rückwärtskante im CFG
- Bearbeitung des Schleifenkörpers analog zur IF-Anweisung, **aber**
- Bei Eintragen einer neuen Phi-Funktion in Kopfknoten
- ... entsteht **neuer** aktueller Wert
- Alle Benutzungen der Variable im Schleifenkörper durch aktuellen Wert ersetzen
 - Verwalte Liste aller im Schleifenkörper benutzten Werte
 - Sogenannte *use chain*
 - Kann für schnelle Korrektur (ändern der Versionsnummer) benutzt werden

OptComp

A. Koch

Organisatorisch

Einleitung

SSA
Erzeugen

SSA für
strukturierte
Sprachen

Dominatoren

Rückwandlung
aus
SSA-Form

Phi-Knoten in WHILE-Anweisungen 1



- Join-Knoten von WHILE-Anweisung ist **Kopfknoten**
 - Zusammentreffen von Schleifeneintritt und Rückwärtskante im CFG
- Bearbeitung des Schleifenkörpers analog zur IF-Anweisung, **aber**
- Bei Eintragen einer neuen Phi-Funktion in Kopfknoten
- ... entsteht **neuer** aktueller Wert
- Alle Benutzungen der Variable im Schleifenkörper durch aktuellen Wert ersetzen
 - Verwalte Liste aller im Schleifenkörper benutzten Werte
 - Sogenannte *use chain*
 - Kann für schnelle Korrektur (ändern der Versionsnummer) benutzt werden

OptComp

A. Koch

Organisatorisch

Einleitung

SSA
Erzeugen

SSA für
strukturierte
Sprachen

Dominatoren

Rückwandlung
aus
SSA-Form

Phi-Knoten in WHILE-Anweisungen 1



- Join-Knoten von WHILE-Anweisung ist **Kopfknoten**
 - Zusammentreffen von Schleifeneintritt und Rückwärtskante im CFG
- Bearbeitung des Schleifenkörpers analog zur IF-Anweisung, **aber**
- Bei Eintragen einer neuen Phi-Funktion in Kopfknoten
- ... entsteht **neuer** aktueller Wert
- Alle Benutzungen der Variable im Schleifenkörper durch aktuellen Wert ersetzen
 - Verwalte Liste aller im Schleifenkörper benutzten Werte
 - Sogenannte *use chain*
 - Kann für schnelle Korrektur (ändern der Versionsnummer) benutzt werden

OptComp

A. Koch

Organisatorisch

Einleitung

SSA
Erzeugen

SSA für
strukturierte
Sprachen

Dominatoren

Rückwandlung
aus
SSA-Form

Phi-Knoten in WHILE-Anweisungen 1



- Join-Knoten von WHILE-Anweisung ist **Kopfknoten**
 - Zusammentreffen von Schleifeneintritt und Rückwärtskante im CFG
- Bearbeitung des Schleifenkörpers analog zur IF-Anweisung, **aber**
- Bei Eintragen einer neuen Phi-Funktion in Kopfknoten
- ... entsteht **neuer** aktueller Wert
- Alle Benutzungen der Variable im Schleifenkörper durch aktuellen Wert ersetzen
 - Verwalte Liste aller im Schleifenkörper benutzten Werte
 - Sogenannte *use chain*
 - Kann für schnelle Korrektur (ändern der Versionsnummer) benutzt werden

OptComp

A. Koch

Organisatorisch

Einleitung

SSA
Erzeugen

SSA für
strukturierte
Sprachen

Dominatoren

Rückwandlung
aus
SSA-Form

Phi-Knoten in WHILE-Anweisungen 1



- Join-Knoten von WHILE-Anweisung ist **Kopfknoten**
 - Zusammentreffen von Schleifeneintritt und Rückwärtskante im CFG
- Bearbeitung des Schleifenkörpers analog zur IF-Anweisung, **aber**
- Bei Eintragen einer neuen Phi-Funktion in Kopfknoten
- ... entsteht **neuer** aktueller Wert
- Alle Benutzungen der Variable im Schleifenkörper durch aktuellen Wert ersetzen
 - Verwalte Liste aller im Schleifenkörper benutzten Werte
 - Sogenannte *use chain*
 - Kann für schnelle Korrektur (ändern der Versionsnummer) benutzt werden

OptComp

A. Koch

Organisatorisch

Einleitung

SSA
Erzeugen

SSA für
strukturierte
Sprachen

Dominatoren

Rückwandlung
aus
SSA-Form

Phi-Knoten in WHILE-Anweisungen 1



- Join-Knoten von WHILE-Anweisung ist **Kopfknoten**
 - Zusammentreffen von Schleifeneintritt und Rückwärtskante im CFG
- Bearbeitung des Schleifenkörpers analog zur IF-Anweisung, **aber**
- Bei Eintragen einer neuen Phi-Funktion in Kopfknoten
- ... entsteht **neuer** aktueller Wert
- Alle Benutzungen der Variable im Schleifenkörper durch aktuellen Wert ersetzen
 - Verwalte Liste aller im Schleifenkörper benutzten Werte
 - Sogenannte *use chain*
 - Kann für schnelle Korrektur (ändern der Versionsnummer) benutzt werden

OptComp

A. Koch

Organisatorisch

Einleitung

SSA
Erzeugen

SSA für
strukturierte
Sprachen

Dominatoren

Rückwandlung
aus
SSA-Form

Phi-Knoten in WHILE-Anweisungen 1



- Join-Knoten von WHILE-Anweisung ist **Kopfknoten**
 - Zusammentreffen von Schleifeneintritt und Rückwärtskante im CFG
- Bearbeitung des Schleifenkörpers analog zur IF-Anweisung, **aber**
- Bei Eintragen einer neuen Phi-Funktion in Kopfknoten
- ... entsteht **neuer** aktueller Wert
- Alle Benutzungen der Variable im Schleifenkörper durch aktuellen Wert ersetzen
 - Verwalte Liste aller im Schleifenkörper benutzten Werte
 - Sogenannte *use chain*
 - Kann für schnelle Korrektur (ändern der Versionsnummer) benutzt werden

OptComp

A. Koch

Organisatorisch

Einleitung

SSA
Erzeugen

SSA für
strukturierte
Sprachen

Dominatoren

Rückwandlung
aus
SSA-Form

Phi-Knoten in WHILE-Anweisungen 1



- Join-Knoten von WHILE-Anweisung ist **Kopfknoten**
 - Zusammentreffen von Schleifeneintritt und Rückwärtskante im CFG
- Bearbeitung des Schleifenkörpers analog zur IF-Anweisung, **aber**
- Bei Eintragen einer neuen Phi-Funktion in Kopfknoten
- ... entsteht **neuer** aktueller Wert
- Alle Benutzungen der Variable im Schleifenkörper durch aktuellen Wert ersetzen
 - Verwalte Liste aller im Schleifenkörper benutzten Werte
 - Sogenannte *use chain*
 - Kann für schnelle Korrektur (ändern der Versionsnummer) benutzt werden

OptComp

A. Koch

Organisatorisch

Einleitung

SSA
Erzeugen

SSA für
strukturierte
Sprachen

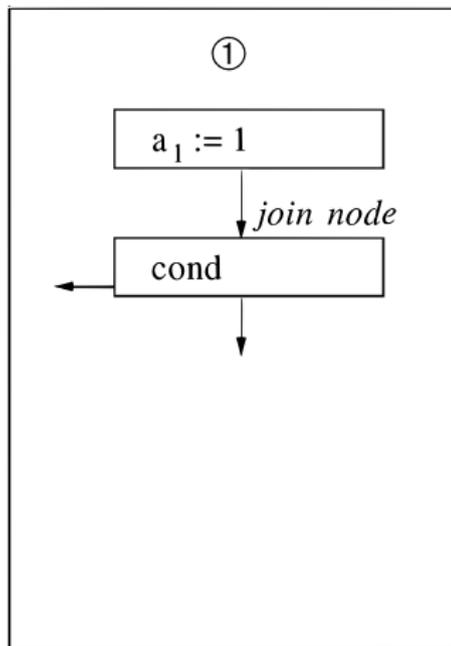
Dominatoren

Rückwandlung
aus
SSA-Form

Phi-Knoten für WHILE-Anweisungen 2



①
a := 1; WHILE cond DO ② b := a + 1; ③ a := a * 2 END



OptComp

A. Koch

Organisatorisch

Einleitung

SSA
Erzeugen

SSA für
strukturierte
Sprachen

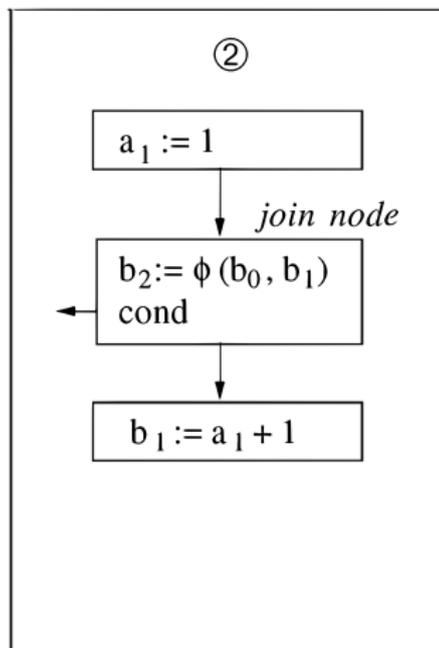
Dominatoren

Rückwandlung
aus
SSA-Form

Phi-Knoten für WHILE-Anweisungen 2



①
a := 1; WHILE cond DO ② b := a + 1; a := a * 2 ③ END



OptComp

A. Koch

Organisatorisch

Einleitung

SSA
Erzeugen

SSA für
strukturierte
Sprachen

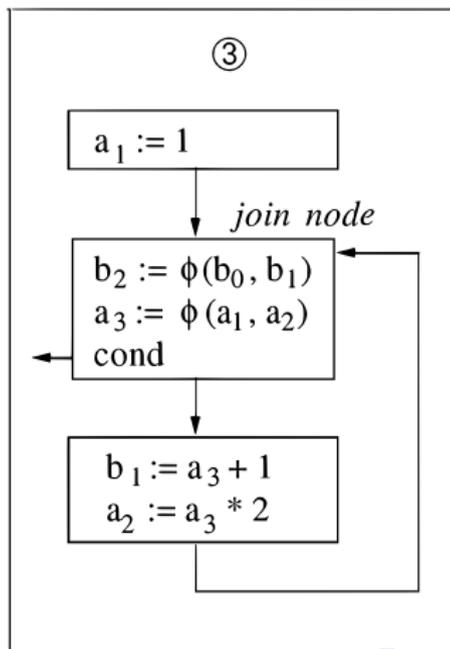
Dominatoren

Rückwandlung
aus
SSA-Form

Phi-Knoten für WHILE-Anweisungen 2



①
a := 1; WHILE cond DO ② b := a + 1; a := a * 2 ③ END



OptComp

A. Koch

Organisatorisch

Einleitung

SSA
Erzeugen

SSA für
strukturierte
Sprachen

Dominatoren

Rückwandlung
aus
SSA-Form

Phi-Knoten für WHILE-Anweisungen 3



OptComp

A. Koch

Organisatorisch

Einleitung

SSA

Erzeugen

SSA für
strukturierte
Sprachen

Dominatoren

Rückwandlung
aus
SSA-Form

- Nach der Bearbeitung des Schleifenkörpers
- ... **Festlegen** der Phi-Zuweisungen im Join-Knoten
- Erzeugt neue Phi-Funktionen in nächstäußerem Join-Knoten
- Legt neue aktuelle Werte für nachfolgende Anweisungen fest
 - Im Beispiel: a_3 und b_2

CASE und FOR würden analog zu IF und WHILE bearbeitet

Phi-Knoten für WHILE-Anweisungen 3



- Nach der Bearbeitung des Schleifenkörpers
- ... **Festlegen** der Phi-Zuweisungen im Join-Knoten
- Erzeugt neue Phi-Funktionen in nächstäußerem Join-Knoten
- Legt neue aktuelle Werte für nachfolgende Anweisungen fest
 - Im Beispiel: a_3 und b_2

CASE und FOR würden analog zu IF und WHILE bearbeitet

OptComp

A. Koch

Organisatorisch

Einleitung

SSA

Erzeugen

SSA für
strukturierte
Sprachen

Dominatoren

Rückwandlung
aus
SSA-Form

Phi-Knoten für WHILE-Anweisungen 3



- Nach der Bearbeitung des Schleifenkörpers
- ... **Festlegen** der Phi-Zuweisungen im Join-Knoten
- Erzeugt neue Phi-Funktionen in nächstäußerem Join-Knoten
- Legt neue aktuelle Werte für nachfolgende Anweisungen fest
 - Im Beispiel: a_3 und b_2

CASE und FOR würden analog zu IF und WHILE bearbeitet

OptComp

A. Koch

Organisatorisch

Einleitung

SSA

Erzeugen

SSA für
strukturierte
Sprachen

Dominatoren

Rückwandlung
aus
SSA-Form

Phi-Knoten für WHILE-Anweisungen 3



- Nach der Bearbeitung des Schleifenkörpers
- ... **Festlegen** der Phi-Zuweisungen im Join-Knoten
- Erzeugt neue Phi-Funktionen in nächstäußerem Join-Knoten
- Legt neue aktuelle Werte für nachfolgende Anweisungen fest
 - Im Beispiel: a_3 und b_2

CASE und FOR würden analog zu IF und WHILE bearbeitet

OptComp

A. Koch

Organisatorisch

Einleitung

SSA

Erzeugen

SSA für
strukturierte
Sprachen

Dominatoren

Rückwandlung
aus
SSA-Form

Phi-Knoten für WHILE-Anweisungen 3



- Nach der Bearbeitung des Schleifenkörpers
- ... **Festlegen** der Phi-Zuweisungen im Join-Knoten
- Erzeugt neue Phi-Funktionen in nächstäußerem Join-Knoten
- Legt neue aktuelle Werte für nachfolgende Anweisungen fest
 - Im Beispiel: a_3 und b_2

CASE und FOR würden analog zu IF und WHILE bearbeitet

OptComp

A. Koch

Organisatorisch

Einleitung

SSA

Erzeugen

SSA für
strukturierte
Sprachen

Dominatoren

Rückwandlung
aus
SSA-Form

Phi-Knoten für WHILE-Anweisungen 3



OptComp

A. Koch

Organisatorisch

Einleitung

SSA

Erzeugen

SSA für
strukturierte
Sprachen

Dominatoren

Rückwandlung
aus
SSA-Form

- Nach der Bearbeitung des Schleifenkörpers
- ... **Festlegen** der Phi-Zuweisungen im Join-Knoten
- Erzeugt neue Phi-Funktionen in nächstäußerem Join-Knoten
- Legt neue aktuelle Werte für nachfolgende Anweisungen fest
 - Im Beispiel: a_3 und b_2

CASE und FOR würden analog zu IF und WHILE bearbeitet

Phi-Knoten für WHILE-Anweisungen 3



OptComp

A. Koch

Organisatorisch

Einleitung

SSA

Erzeugen

SSA für
strukturierte
Sprachen

Dominatoren

Rückwandlung
aus
SSA-Form

- Nach der Bearbeitung des Schleifenkörpers
- ... **Festlegen** der Phi-Zuweisungen im Join-Knoten
- Erzeugt neue Phi-Funktionen in nächstäußerem Join-Knoten
- Legt neue aktuelle Werte für nachfolgende Anweisungen fest
 - Im Beispiel: a_3 und b_2

CASE und FOR würden analog zu IF und WHILE bearbeitet

Phi-Knoten für REPEAT-Anweisungen 1



- **Sonderfall!**
- Konstrukt wird **nicht** über Join-Knoten verlassen
- Analog zu WHILE: Join-Knoten ist Schleifenkopf
 - Hier auch Phi-Zuweisungen untergebracht
- Aber Unterschied beim **Festlegen** des Join-Blocks!
- Aktueller Wert ist **nicht** Ziel der Phi-Zuweisung (wie bei WHILE)
- ... sondern Wert zugeordnet der **Rückwärtskante**
 - Sonst wären Änderungen nach einem Schleifendurchlauf nicht sichtbar
- Gleichen Wert auch für Operanden **nächstäußerer** Phi-Funktion verwenden

OptComp

A. Koch

Organisatorisch

Einleitung

SSA

Erzeugen

SSA für
strukturierte
Sprachen

Dominatoren

Rückwandlung
aus
SSA-Form

Phi-Knoten für REPEAT-Anweisungen 1



- Sonderfall!
- Konstrukt wird **nicht** über Join-Knoten verlassen
- Analog zu WHILE: Join-Knoten ist Schleifenkopf
 - Hier auch Phi-Zuweisungen untergebracht
- Aber Unterschied beim **Festlegen** des Join-Blocks!
- Aktueller Wert ist **nicht** Ziel der Phi-Zuweisung (wie bei WHILE)
- ... sondern Wert zugeordnet der **Rückwärtskante**
 - Sonst wären Änderungen nach einem Schleifendurchlauf nicht sichtbar
- Gleichen Wert auch für Operanden **nächstäußerer** Phi-Funktion verwenden

OptComp

A. Koch

Organisatorisch

Einleitung

SSA

Erzeugen

SSA für
strukturierte
Sprachen

Dominatoren

Rückwandlung
aus
SSA-Form

Phi-Knoten für REPEAT-Anweisungen 1



- Sonderfall!
- Konstrukt wird **nicht** über Join-Knoten verlassen
- Analog zu WHILE: Join-Knoten ist Schleifenkopf
 - Hier auch Phi-Zuweisungen untergebracht
- Aber Unterschied beim **Festlegen** des Join-Blocks!
- Aktueller Wert ist **nicht** Ziel der Phi-Zuweisung (wie bei WHILE)
- ... sondern Wert zugeordnet der **Rückwärtskante**
 - Sonst wären Änderungen nach einem Schleifendurchlauf nicht sichtbar
- Gleichen Wert auch für Operanden **nächstäußerer** Phi-Funktion verwenden

OptComp

A. Koch

Organisatorisch

Einleitung

SSA

Erzeugen

SSA für
strukturierte
Sprachen

Dominatoren

Rückwandlung
aus
SSA-Form

Phi-Knoten für REPEAT-Anweisungen 1



- Sonderfall!
- Konstrukt wird **nicht** über Join-Knoten verlassen
- Analog zu WHILE: Join-Knoten ist Schleifenkopf
 - Hier auch Phi-Zuweisungen untergebracht
- Aber Unterschied beim **Festlegen** des Join-Blocks!
- Aktueller Wert ist **nicht** Ziel der Phi-Zuweisung (wie bei WHILE)
- ... sondern Wert zugeordnet der **Rückwärtskante**
 - Sonst wären Änderungen nach einem Schleifendurchlauf nicht sichtbar
- Gleichen Wert auch für Operanden **nächstäußerer** Phi-Funktion verwenden

OptComp

A. Koch

Organisatorisch

Einleitung

SSA

Erzeugen

SSA für
strukturierte
Sprachen

Dominatoren

Rückwandlung
aus
SSA-Form

Phi-Knoten für REPEAT-Anweisungen 1



- Sonderfall!
- Konstrukt wird **nicht** über Join-Knoten verlassen
- Analog zu WHILE: Join-Knoten ist Schleifenkopf
 - Hier auch Phi-Zuweisungen untergebracht
- Aber Unterschied beim **Festlegen** des Join-Blocks!
- Aktueller Wert ist **nicht** Ziel der Phi-Zuweisung (wie bei WHILE)
- ... sondern Wert zugeordnet der **Rückwärtskante**
 - Sonst wären Änderungen nach einem Schleifendurchlauf nicht sichtbar
- Gleichen Wert auch für Operanden **nächstäußerer** Phi-Funktion verwenden

OptComp

A. Koch

Organisatorisch

Einleitung

SSA

Erzeugen

SSA für
strukturierte
Sprachen

Dominatoren

Rückwandlung
aus
SSA-Form

Phi-Knoten für REPEAT-Anweisungen 1



- Sonderfall!
- Konstrukt wird **nicht** über Join-Knoten verlassen
- Analog zu WHILE: Join-Knoten ist Schleifenkopf
 - Hier auch Phi-Zuweisungen untergebracht
- Aber Unterschied beim **Festlegen** des Join-Blocks!
- Aktueller Wert ist **nicht** Ziel der Phi-Zuweisung (wie bei WHILE)
- ... sondern Wert zugeordnet der **Rückwärtskante**
 - Sonst wären Änderungen nach einem Schleifendurchlauf nicht sichtbar
- Gleichen Wert auch für Operanden **nächstäußerer** Phi-Funktion verwenden

OptComp

A. Koch

Organisatorisch

Einleitung

SSA

Erzeugen

SSA für
strukturierte
Sprachen

Dominatoren

Rückwandlung
aus
SSA-Form

Phi-Knoten für REPEAT-Anweisungen 1



- Sonderfall!
- Konstrukt wird **nicht** über Join-Knoten verlassen
- Analog zu WHILE: Join-Knoten ist Schleifenkopf
 - Hier auch Phi-Zuweisungen untergebracht
- Aber Unterschied beim **Festlegen** des Join-Blocks!
- Aktueller Wert ist **nicht** Ziel der Phi-Zuweisung (wie bei WHILE)
- ... sondern Wert zugeordnet der **Rückwärtskante**
 - Sonst wären Änderungen nach einem Schleifendurchlauf nicht sichtbar
- Gleichen Wert auch für Operanden **nächstäußerer** Phi-Funktion verwenden

OptComp

A. Koch

Organisatorisch

Einleitung

SSA

Erzeugen

SSA für
strukturierte
Sprachen

Dominatoren

Rückwandlung
aus
SSA-Form

Phi-Knoten für REPEAT-Anweisungen 1



- Sonderfall!
- Konstrukt wird **nicht** über Join-Knoten verlassen
- Analog zu WHILE: Join-Knoten ist Schleifenkopf
 - Hier auch Phi-Zuweisungen untergebracht
- Aber Unterschied beim **Festlegen** des Join-Blocks!
- Aktueller Wert ist **nicht** Ziel der Phi-Zuweisung (wie bei WHILE)
- ... sondern Wert zugeordnet der **Rückwärtskante**
 - Sonst wären Änderungen nach einem Schleifendurchlauf nicht sichtbar
- Gleichen Wert auch für Operanden **nächstäußerer** Phi-Funktion verwenden

OptComp

A. Koch

Organisatorisch

Einleitung

SSA

Erzeugen

SSA für
strukturierte
Sprachen

Dominatoren

Rückwandlung
aus
SSA-Form

Phi-Knoten für REPEAT-Anweisungen 1



- Sonderfall!
- Konstrukt wird **nicht** über Join-Knoten verlassen
- Analog zu WHILE: Join-Knoten ist Schleifenkopf
 - Hier auch Phi-Zuweisungen untergebracht
- Aber Unterschied beim **Festlegen** des Join-Blocks!
- Aktueller Wert ist **nicht** Ziel der Phi-Zuweisung (wie bei WHILE)
- ... sondern Wert zugeordnet der **Rückwärtskante**
 - Sonst wären Änderungen nach einem Schleifendurchlauf nicht sichtbar
- Gleichen Wert auch für Operanden **nächstäußerer** Phi-Funktion verwenden

OptComp

A. Koch

Organisatorisch

Einleitung

SSA

Erzeugen

SSA für
strukturierte
Sprachen

Dominatoren

Rückwandlung
aus
SSA-Form

Phi-Knoten für REPEAT-Anweisungen 2



OptComp

A. Koch

Organisatorisch

Einleitung

SSA

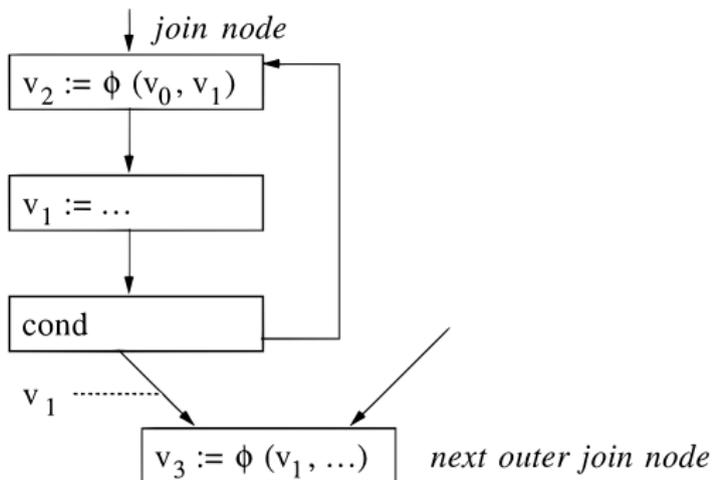
Erzeugen

SSA für
strukturierte
Sprachen

Dominatoren

Rückwandlung
aus
SSA-Form

REPEAT
 $v := \dots$
UNTIL cond



Beachte: Weiterverwendung von v_1 , **nicht** von v_2



● INSERTPHI

- Erzeugt neue oder modifiziert bestehende Phi-Zuweisung in Join-Knoten b
- Aufruf: $\text{INSERTPHI}(b, i, v_i, v_{old})$
 - Zur Bearbeitung von Zuweisung $v_i := \dots$
 - \dots die im i -ten, zum Block b führenden Zweig steht
 - v_{old} ist aktueller Wert vor dieser Zuweisung
 - Wird als Sicherheitskopie abgespeichert

● COMMITPHI

- Legt die Phi-Zuweisungen in einem Join-Knoten b fest
- Bestimmt aktuelle Werte
- Propagiert neue Phi-Zuweisungen in nächstäußeren Join-Knoten B , über die Kante I kommend



● INSERTPHI

- Erzeugt neue oder modifiziert bestehende Phi-Zuweisung in Join-Knoten b
- Aufruf: $\text{INSERTPHI}(b, i, v_i, v_{old})$
 - Zur Bearbeitung von Zuweisung $v_i := \dots$
 - ... die im i -ten, zum Block b führenden Zweig steht
 - v_{old} ist aktueller Wert vor dieser Zuweisung
 - Wird als Sicherheitskopie abgespeichert

● COMMITPHI

- Legt die Phi-Zuweisungen in einem Join-Knoten b fest
- Bestimmt aktuelle Werte
- Propagiert neue Phi-Zuweisungen in nächstäußeren Join-Knoten B , über die Kante I kommend



● INSERTPHI

- Erzeugt neue oder modifiziert bestehende Phi-Zuweisung in Join-Knoten b
- Aufruf: $\text{INSERTPHI}(b, i, v_i, v_{old})$
 - Zur Bearbeitung von Zuweisung $v_i := \dots$
 - ...die im i -ten, zum Block b führenden Zweig steht
 - v_{old} ist aktueller Wert **vor** dieser Zuweisung
 - Wird als Sicherheitskopie abgespeichert

● COMMITPHI

- Legt die Phi-Zuweisungen in einem Join-Knoten b fest
- Bestimmt aktuelle Werte
- Propagiert neue Phi-Zuweisungen in nächstäußeren Join-Knoten B , über die Kante I kommend



● INSERTPHI

- Erzeugt neue oder modifiziert bestehende Phi-Zuweisung in Join-Knoten b
- Aufruf: $\text{INSERTPHI}(b, i, v_i, v_{old})$
 - Zur Bearbeitung von Zuweisung $v_i := \dots$
 - \dots die im i -ten, zum Block b führenden Zweig steht
 - v_{old} ist aktueller Wert vor dieser Zuweisung
 - Wird als Sicherheitskopie abgespeichert

● COMMITPHI

- Legt die Phi-Zuweisungen in einem Join-Knoten b fest
- Bestimmt aktuelle Werte
- Propagiert neue Phi-Zuweisungen in nächstäußeren Join-Knoten B , über die Kante I kommend



● INSERTPHI

- Erzeugt neue oder modifiziert bestehende Phi-Zuweisung in Join-Knoten b
- Aufruf: $\text{INSERTPHI}(b, i, v_i, v_{old})$
 - Zur Bearbeitung von Zuweisung $v_i := \dots$
 - ...die im i -ten, zum Block b führenden Zweig steht
 - v_{old} ist aktueller Wert vor dieser Zuweisung
 - Wird als Sicherheitskopie abgespeichert

● COMMITPHI

- Legt die Phi-Zuweisungen in einem Join-Knoten b fest
- Bestimmt aktuelle Werte
- Propagiert neue Phi-Zuweisungen in nächstäußeren Join-Knoten B , über die Kante I kommend



● INSERTPHI

- Erzeugt neue oder modifiziert bestehende Phi-Zuweisung in Join-Knoten b
- Aufruf: $\text{INSERTPHI}(b, i, v_i, v_{old})$
 - Zur Bearbeitung von Zuweisung $v_i := \dots$
 - ...die im i -ten, zum Block b führenden Zweig steht
 - v_{old} ist aktueller Wert **vor** dieser Zuweisung
 - Wird als Sicherheitskopie abgespeichert

● COMMITPHI

- Legt die Phi-Zuweisungen in einem Join-Knoten b fest
- Bestimmt aktuelle Werte
- Propagiert neue Phi-Zuweisungen in nächstäußeren Join-Knoten B , über die Kante I kommend



● INSERTPHI

- Erzeugt neue oder modifiziert bestehende Phi-Zuweisung in Join-Knoten b
- Aufruf: $\text{INSERTPHI}(b, i, v_i, v_{old})$
 - Zur Bearbeitung von Zuweisung $v_i := \dots$
 - \dots die im i -ten, zum Block b führenden Zweig steht
 - v_{old} ist aktueller Wert **vor** dieser Zuweisung
 - Wird als Sicherheitskopie abgespeichert

● COMMITPHI

- Legt die Phi-Zuweisungen in einem Join-Knoten b fest
- Bestimmt aktuelle Werte
- Propagiert neue Phi-Zuweisungen in nächstäußeren Join-Knoten B , über die Kante I kommend



● INSERTPHI

- Erzeugt neue oder modifiziert bestehende Phi-Zuweisung in Join-Knoten b
- Aufruf: $\text{INSERTPHI}(b, i, v_i, v_{old})$
 - Zur Bearbeitung von Zuweisung $v_i := \dots$
 - ...die im i -ten, zum Block b führenden Zweig steht
 - v_{old} ist aktueller Wert **vor** dieser Zuweisung
 - Wird als Sicherheitskopie abgespeichert

● COMMITPHI

- Legt die Phi-Zuweisungen in einem Join-Knoten b fest
- Bestimmt aktuelle Werte
- Propagiert neue Phi-Zuweisungen in nächstäußeren Join-Knoten B , über die Kante I kommend



● INSERTPHI

- Erzeugt neue oder modifiziert bestehende Phi-Zuweisung in Join-Knoten b
- Aufruf: $\text{INSERTPHI}(b, i, v_i, v_{old})$
 - Zur Bearbeitung von Zuweisung $v_i := \dots$
 - \dots die im i -ten, zum Block b führenden Zweig steht
 - v_{old} ist aktueller Wert **vor** dieser Zuweisung
 - Wird als Sicherheitskopie abgespeichert

● COMMITPHI

- Legt die Phi-Zuweisungen in einem Join-Knoten b fest
- Bestimmt aktuelle Werte
- Propagiert neue Phi-Zuweisungen in nächstäußeren Join-Knoten B , über die Kante I kommend



● INSERTPHI

- Erzeugt neue oder modifiziert bestehende Phi-Zuweisung in Join-Knoten b
- Aufruf: $\text{INSERTPHI}(b, i, v_i, v_{old})$
 - Zur Bearbeitung von Zuweisung $v_i := \dots$
 - \dots die im i -ten, zum Block b führenden Zweig steht
 - v_{old} ist aktueller Wert **vor** dieser Zuweisung
 - Wird als Sicherheitskopie abgespeichert

● COMMITPHI

- Legt die Phi-Zuweisungen in einem Join-Knoten b fest
- Bestimmt aktuelle Werte
- Propagiert neue Phi-Zuweisungen in nächstäußeren Join-Knoten B , über die Kante I kommend



● INSERTPHI

- Erzeugt neue oder modifiziert bestehende Phi-Zuweisung in Join-Knoten b
- Aufruf: $\text{INSERTPHI}(b, i, v_i, v_{old})$
 - Zur Bearbeitung von Zuweisung $v_i := \dots$
 - \dots die im i -ten, zum Block b führenden Zweig steht
 - v_{old} ist aktueller Wert **vor** dieser Zuweisung
 - Wird als Sicherheitskopie abgespeichert

● COMMITPHI

- Legt die Phi-Zuweisungen in einem Join-Knoten b fest
- Bestimmt aktuelle Werte
- Propagiert neue Phi-Zuweisungen in nächstäußeren Join-Knoten B , über die Kante I kommend



```
PROCEDURE InsertPhi (b: Node; i: INTEGER; vi, vold: Value);
BEGIN
  IF b contains no  $\phi$ -assignment for v THEN
    Insert " $v_j := \phi(v_{old}, \dots, v_{old}) / v_{old}$ " in b;
    IF b is a join node of a loop THEN
      Rename all mentions of vold in the loop to vj
    END
  END;
  Replace i-th operand of v's  $\phi$ -assignment by vj
END InsertPhi;
```



```
PROCEDURE CommitPhi (b: Node);
BEGIN
  FOR all  $\phi$ -instructions " $v_i := \phi(v_0, \dots, v_n) / v_{old}$ " in b DO
    IF  $b$  is a join node of a repeat THEN val :=  $v_n$  ELSE val :=  $v_i$  END;
    Make val the current value of  $v$ ;
    InsertPhi(B, I, val,  $v_{old}$ )
  END
END CommitPhi;
```

Hier Annahme: Letzter Zweig n ist Rückwärtskante der REPEAT-Schleife



```
PROCEDURE CommitPhi (b: Node);
BEGIN
  FOR all  $\phi$ -instructions " $v_i := \phi(v_0, \dots, v_n) / v_{old}$ " in b DO
    IF  $b$  is a join node of a repeat THEN val :=  $v_n$  ELSE val :=  $v_i$  END;
    Make val the current value of  $v$ ;
    InsertPhi(B, l, val,  $v_{old}$ )
  END
END CommitPhi;
```

Hier Annahme: Letzter Zweig n ist Rückwärtskante der REPEAT-Schleife



- Hier nicht gezeigt: Rücksetzen auf v_{old} bei Bearbeitung des nächsten Zweiges
- Variablen in Triangle durch Verweise auf Definitionen kennzeichnen
- Keine String-Vergleiche mehr nötig!
- Werte sind dann Tupel (Definition, Versionsnummer)



- Hier nicht gezeigt: Rücksetzen auf v_{old} bei Bearbeitung des nächsten Zweiges
- Variablen in Triangle durch Verweise auf Definitionen kennzeichnen
- Keine String-Vergleiche mehr nötig!
- Werte sind dann Tupel (Definition, Versionsnummer)



- Hier nicht gezeigt: Rücksetzen auf v_{old} bei Bearbeitung des nächsten Zweiges
- Variablen in Triangle durch Verweise auf Definitionen kennzeichnen
- Keine String-Vergleiche mehr nötig!
- Werte sind dann Tupel (Definition, Versionsnummer)



- Hier nicht gezeigt: Rücksetzen auf v_{old} bei Bearbeitung des nächsten Zweiges
- Variablen in Triangle durch Verweise auf Definitionen kennzeichnen
- Keine String-Vergleiche mehr nötig!
- Werte sind dann Tupel (Definition, Versionsnummer)

Berechnung von Dominatoren 1



OptComp

A. Koch

Organisatorisch

Einleitung

SSA
Erzeugen

SSA für
strukturierte
Sprachen

Dominatoren

Rückwandlung
aus
SSA-Form

- Muß bei Cytron et al. bei der SSA-Umformung gemacht werden
- War hier nicht nötig
- Dominatoren sind aber nach wie vor nützlich
- Wie sind sie hier berechenbar?
- Viel einfacher als im allgemeinen Fall!

Berechnung von Dominatoren 1



OptComp

A. Koch

Organisatorisch

Einleitung

SSA
Erzeugen

SSA für
strukturierte
Sprachen

Dominatoren

Rückwandlung
aus
SSA-Form

- Muß bei Cytron et al. bei der SSA-Umformung gemacht werden
- War hier nicht nötig
- Dominatoren sind aber nach wie vor nützlich
- Wie sind sie hier berechenbar?
- Viel einfacher als im allgemeinen Fall!

Berechnung von Dominatoren 1



OptComp

A. Koch

Organisatorisch

Einleitung

SSA
Erzeugen

SSA für
strukturierte
Sprachen

Dominatoren

Rückwandlung
aus
SSA-Form

- Muß bei Cytron et al. bei der SSA-Umformung gemacht werden
- War hier nicht nötig
- Dominatoren sind aber nach wie vor nützlich
- Wie sind sie hier berechenbar?
- Viel einfacher als im allgemeinen Fall!



- Muß bei Cytron et al. bei der SSA-Umformung gemacht werden
- War hier nicht nötig
- Dominatoren sind aber nach wie vor nützlich
- Wie sind sie hier berechenbar?
- Viel einfacher als im allgemeinen Fall!



- Muß bei Cytron et al. bei der SSA-Umformung gemacht werden
- War hier nicht nötig
- Dominatoren sind aber nach wie vor nützlich
- Wie sind sie hier berechenbar?
- Viel einfacher als im allgemeinen Fall!

Berechnung von Dominatoren 2



- Auch hier Berechnung in einem Pass möglich
 - Über Quelltext oder AST
- Erinnerung: Dominatorbaum
 - Vater eines Blocks ist dessen unmittelbarer Dominator
IDOM
- Idee hier: Sub-CFGs der Konstrukte
IF/WHILE/FOR/REPEAT/CASE
- ... haben **einen** Eintrittspunkt und **einen** Austrittspunkt
- Der Eintrittspunkt dominiert **alle** Knoten des Konstrukts
- Unmittelbare Dominatoren können immer nach dem gleichen Schema bestimmt werden
- Dann Hochhangeln für weiter entfernte Dominatoren

OptComp

A. Koch

Organisatorisch

Einleitung

SSA
Erzeugen

SSA für
strukturierte
Sprachen

Dominatoren

Rückwandlung
aus
SSA-Form

Berechnung von Dominatoren 2



- Auch hier Berechnung in einem Pass möglich
 - Über Quelltext oder AST
- Erinnerung: Dominatorbaum
 - Vater eines Blocks ist dessen unmittelbarer Dominator
IDOM
- Idee hier: Sub-CFGs der Konstrukte
IF/WHILE/FOR/REPEAT/CASE
- ... haben **einen** Eintrittspunkt und **einen** Austrittspunkt
- Der Eintrittspunkt dominiert **alle** Knoten des Konstrukts
- Unmittelbare Dominatoren können immer nach dem gleichen Schema bestimmt werden
- Dann Hochhangeln für weiter entfernte Dominatoren

OptComp

A. Koch

Organisatorisch

Einleitung

SSA
Erzeugen

SSA für
strukturierte
Sprachen

Dominatoren

Rückwandlung
aus
SSA-Form

Berechnung von Dominatoren 2



- Auch hier Berechnung in einem Pass möglich
 - Über Quelltext oder AST
- Erinnerung: Dominatorbaum
 - Vater eines Blocks ist dessen unmittelbarer Dominator
IDOM
- Idee hier: Sub-CFGs der Konstrukte
IF/WHILE/FOR/REPEAT/CASE
- ... haben **einen** Eintrittspunkt und **einen** Austrittspunkt
- Der Eintrittspunkt dominiert **alle** Knoten des Konstrukts
- Unmittelbare Dominatoren können immer nach dem gleichen Schema bestimmt werden
- Dann Hochhangeln für weiter entfernte Dominatoren

OptComp

A. Koch

Organisatorisch

Einleitung

SSA
Erzeugen

SSA für
strukturierte
Sprachen

Dominatoren

Rückwandlung
aus
SSA-Form

Berechnung von Dominatoren 2



- Auch hier Berechnung in einem Pass möglich
 - Über Quelltext oder AST
- Erinnerung: Dominatorbaum
 - Vater eines Blocks ist dessen unmittelbarer Dominator IDOM
- Idee hier: Sub-CFGs der Konstrukte IF/WHILE/FOR/REPEAT/CASE
- ... haben **einen** Eintrittspunkt und **einen** Austrittspunkt
- Der Eintrittspunkt dominiert **alle** Knoten des Konstrukts
- Unmittelbare Dominatoren können immer nach dem gleichen Schema bestimmt werden
- Dann Hochhangeln für weiter entfernte Dominatoren

OptComp

A. Koch

Organisatorisch

Einleitung

SSA

Erzeugen

SSA für
strukturierte
Sprachen

Dominatoren

Rückwandlung
aus
SSA-Form



- Auch hier Berechnung in einem Pass möglich
 - Über Quelltext oder AST
- Erinnerung: Dominatorbaum
 - Vater eines Blocks ist dessen unmittelbarer Dominator IDOM
- Idee hier: Sub-CFGs der Konstrukte IF/WHILE/FOR/REPEAT/CASE
- ... haben **einen** Eintrittspunkt und **einen** Austrittspunkt
- Der Eintrittspunkt dominiert **alle** Knoten des Konstrukts
- Unmittelbare Dominatoren können immer nach dem gleichen Schema bestimmt werden
- Dann Hochhangeln für weiter entfernte Dominatoren

OptComp

A. Koch

Organisatorisch

Einleitung

SSA

Erzeugen

SSA für
strukturierte
Sprachen

Dominatoren

Rückwandlung
aus
SSA-Form

Berechnung von Dominatoren 2



- Auch hier Berechnung in einem Pass möglich
 - Über Quelltext oder AST
- Erinnerung: Dominatorbaum
 - Vater eines Blocks ist dessen unmittelbarer Dominator IDOM
- Idee hier: Sub-CFGs der Konstrukte IF/WHILE/FOR/REPEAT/CASE
- ... haben **einen** Eintrittspunkt und **einen** Austrittspunkt
- Der Eintrittspunkt dominiert **alle** Knoten des Konstrukts
- Unmittelbare Dominatoren können immer nach dem gleichen Schema bestimmt werden
- Dann Hochhangeln für weiter entfernte Dominatoren

OptComp

A. Koch

Organisatorisch

Einleitung

SSA
Erzeugen

SSA für
strukturierte
Sprachen

Dominatoren

Rückwandlung
aus
SSA-Form

Berechnung von Dominatoren 2



- Auch hier Berechnung in einem Pass möglich
 - Über Quelltext oder AST
- Erinnerung: Dominatorbaum
 - Vater eines Blocks ist dessen unmittelbarer Dominator IDOM
- Idee hier: Sub-CFGs der Konstrukte IF/WHILE/FOR/REPEAT/CASE
- ... haben **einen** Eintrittspunkt und **einen** Austrittspunkt
- Der Eintrittspunkt dominiert **alle** Knoten des Konstrukts
- Unmittelbare Dominatoren können immer nach dem gleichen Schema bestimmt werden
- Dann Hochhangeln für weiter entfernte Dominatoren

OptComp

A. Koch

Organisatorisch

Einleitung

SSA
Erzeugen

SSA für
strukturierte
Sprachen

Dominatoren

Rückwandlung
aus
SSA-Form



- Auch hier Berechnung in einem Pass möglich
 - Über Quelltext oder AST
- Erinnerung: Dominatorbaum
 - Vater eines Blocks ist dessen unmittelbarer Dominator IDOM
- Idee hier: Sub-CFGs der Konstrukte IF/WHILE/FOR/REPEAT/CASE
- ... haben **einen** Eintrittspunkt und **einen** Austrittspunkt
- Der Eintrittspunkt dominiert **alle** Knoten des Konstrukts
- Unmittelbare Dominatoren können immer nach dem gleichen Schema bestimmt werden
- Dann Hochhangeln für weiter entfernte Dominatoren

Berechnung von Dominatoren 2



- Auch hier Berechnung in einem Pass möglich
 - Über Quelltext oder AST
- Erinnerung: Dominatorbaum
 - Vater eines Blocks ist dessen unmittelbarer Dominator IDOM
- Idee hier: Sub-CFGs der Konstrukte IF/WHILE/FOR/REPEAT/CASE
- ... haben **einen** Eintrittspunkt und **einen** Austrittspunkt
- Der Eintrittspunkt dominiert **alle** Knoten des Konstrukts
- Unmittelbare Dominatoren können immer nach dem gleichen Schema bestimmt werden
- Dann Hochhangeln für weiter entfernte Dominatoren

OptComp

A. Koch

Organisatorisch

Einleitung

SSA

Erzeugen

SSA für
strukturierte
Sprachen

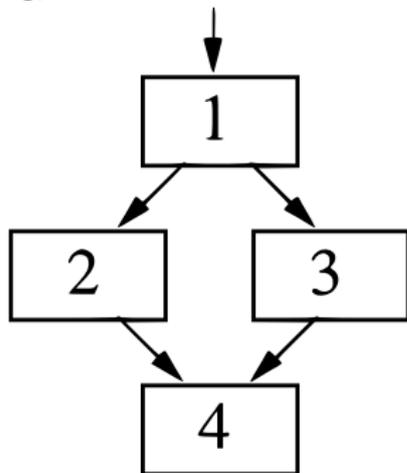
Dominatoren

Rückwandlung
aus
SSA-Form

Berechnung von Dominatoren für IF, CASE



CFG



Dominatorbaum

OptComp

A. Koch

Organisatorisch

Einleitung

SSA
Erzeugen

SSA für
strukturierte
Sprachen

Dominatoren

Rückwandlung
aus
SSA-Form

Berechnung von Dominatoren für IF, CASE



OptComp

A. Koch

Organisatorisch

Einleitung

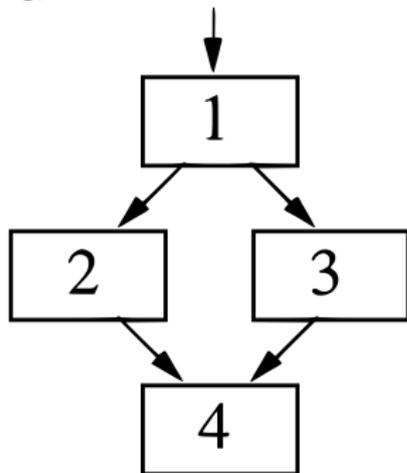
SSA
Erzeugen

SSA für
strukturierte
Sprachen

Dominatoren

Rückwandlung
aus
SSA-Form

CFG

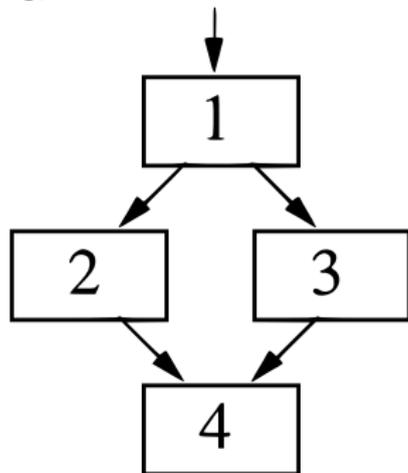


Dominatorbaum

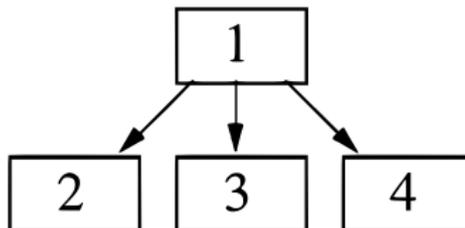
Berechnung von Dominatoren für IF, CASE



CFG



Dominatorbaum



OptComp

A. Koch

Organisatorisch

Einleitung

SSA
Erzeugen

SSA für
strukturierte
Sprachen

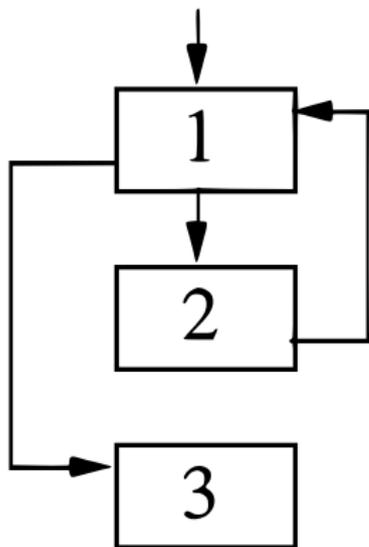
Dominatoren

Rückwandlung
aus
SSA-Form

Berechnung von Dominatoren für WHILE, FOR



CFG



Dominatorbaum

OptComp

A. Koch

Organisatorisch

Einleitung

SSA
Erzeugen

SSA für
strukturierte
Sprachen

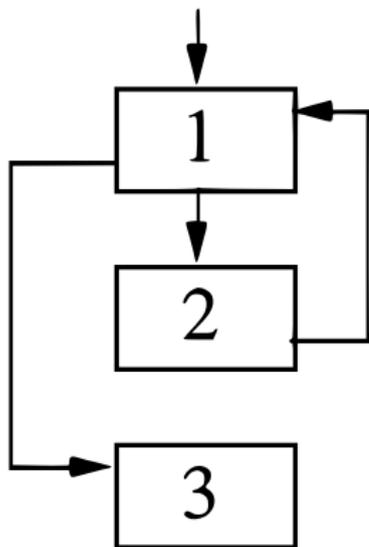
Dominatorn

Rückwandlung
aus
SSA-Form

Berechnung von Dominatoren für WHILE, FOR



CFG



Dominatorbaum

OptComp

A. Koch

Organisatorisch

Einleitung

SSA
Erzeugen

SSA für
strukturierte
Sprachen

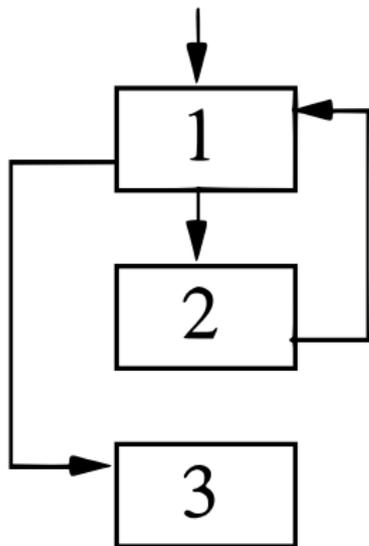
Dominatoren

Rückwandlung
aus
SSA-Form

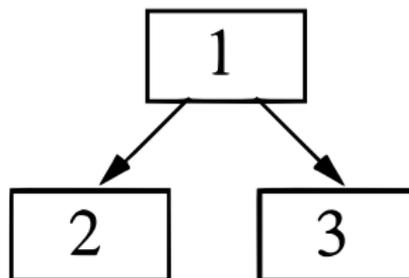
Berechnung von Dominatoren für WHILE, FOR



CFG



Dominatorbaum



OptComp

A. Koch

Organisatorisch

Einleitung

SSA
Erzeugen

SSA für
strukturierte
Sprachen

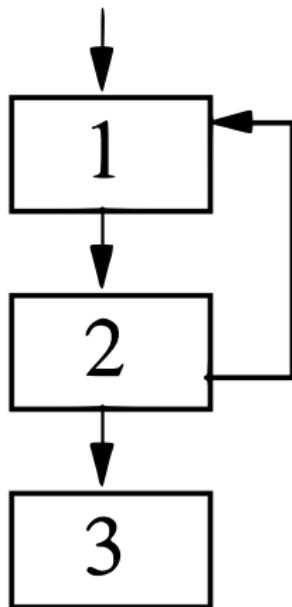
Dominatoren

Rückwandlung
aus
SSA-Form

Berechnung von Dominatoren für REPEAT



CFG



Dominatorbaum

OptComp

A. Koch

Organisatorisch

Einleitung

SSA
Erzeugen

SSA für
strukturierte
Sprachen

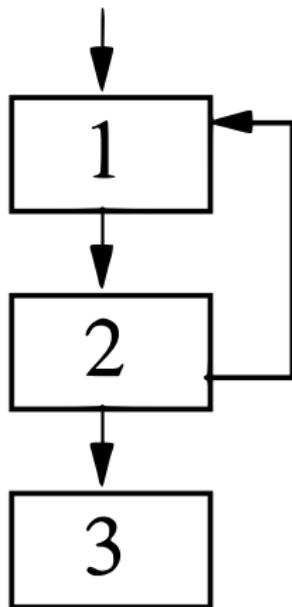
Dominatorn

Rückwandlung
aus
SSA-Form

Berechnung von Dominatoren für REPEAT



CFG



Dominatorbaum

OptComp

A. Koch

Organisatorisch

Einleitung

SSA
Erzeugen

SSA für
strukturierte
Sprachen

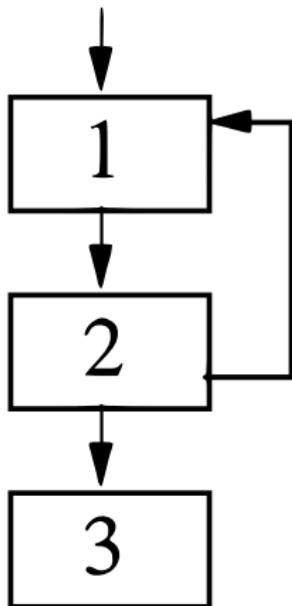
Dominatorn

Rückwandlung
aus
SSA-Form

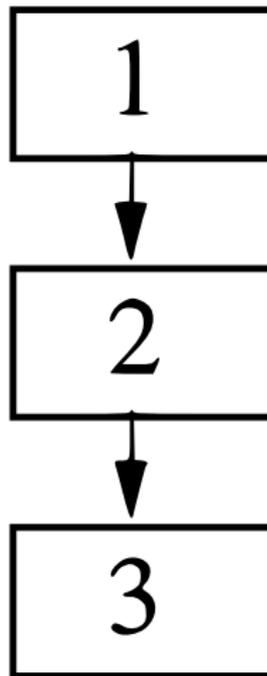
Berechnung von Dominatoren für REPEAT



CFG



Dominatorbaum



OptComp

A. Koch

Organisatorisch

Einleitung

SSA
Erzeugen

SSA für
strukturierte
Sprachen

Dominatoren

Rückwandlung
aus
SSA-Form

Rückwandlung aus der SSA-Form 1



OptComp

A. Koch

Organisatorisch

Einleitung

SSA
Erzeugen

SSA für
strukturierte
Sprachen

Dominatoren

Rückwandlung
aus
SSA-Form

- Normale Prozessoren haben keine Phi-Instruktion
- Phi-Instruktionen müssen entfernt werden

Rückwandlung aus der SSA-Form 1



OptComp

A. Koch

Organisatorisch

Einleitung

SSA
Erzeugen

SSA für
strukturierte
Sprachen

Dominatoren

Rückwandlung
aus
SSA-Form

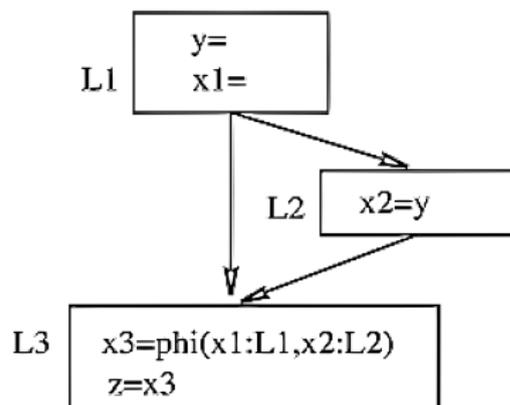
- Normale Prozessoren haben keine Phi-Instruktion
- Phi-Instruktionen müssen entfernt werden

Rückwandlung aus SSA-Form 2



Naive Idee: Phi einfach löschen und Wertnummern entfernen

Vorher:



Nachher:

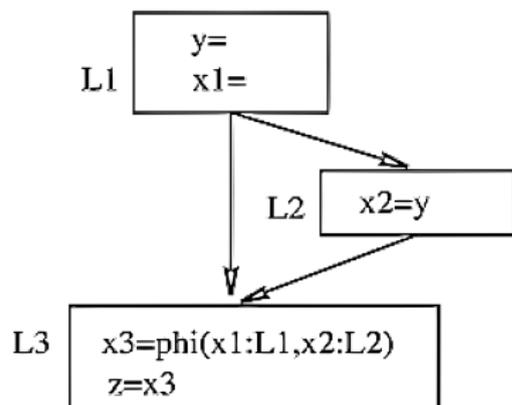
... so weit, so gut.

Rückwandlung aus SSA-Form 2

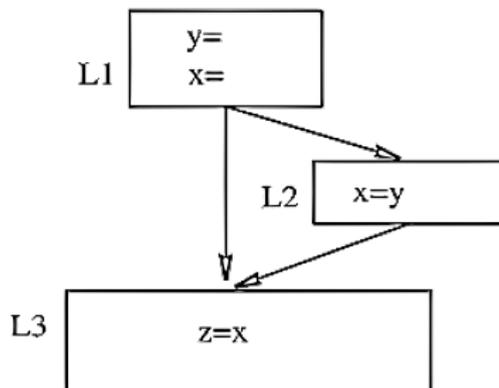


Naive Idee: Phi einfach löschen und Wertnummern entfernen

Vorher:



Nachher:



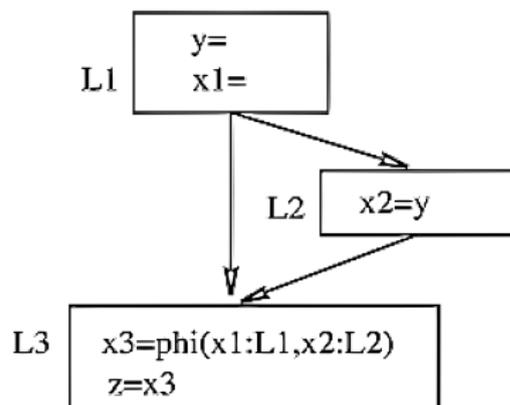
... so weit, so gut.

Rückwandlung aus SSA-Form 2

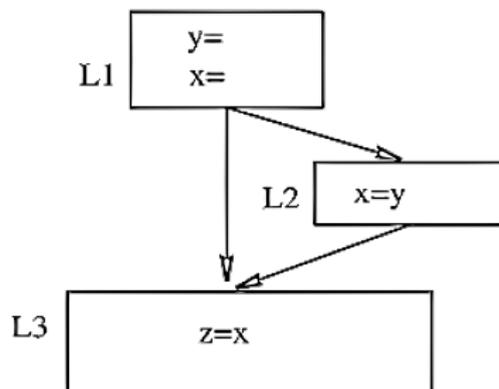


Naive Idee: Phi einfach löschen und Wertnummern entfernen

Vorher:



Nachher:



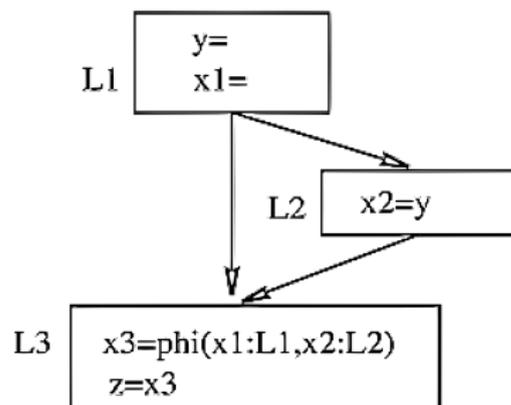
... so weit, so gut.

Rückwandlung aus SSA-Form 2

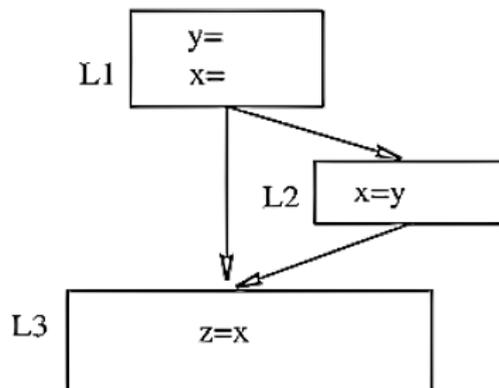


Naive Idee: Phi einfach löschen und Wertnummern entfernen

Vorher:



Nachher:



... so weit, so gut.

OptComp

A. Koch

Organisatorisch

Einleitung

SSA
Erzeugen

SSA für
strukturierte
Sprachen

Dominatoren

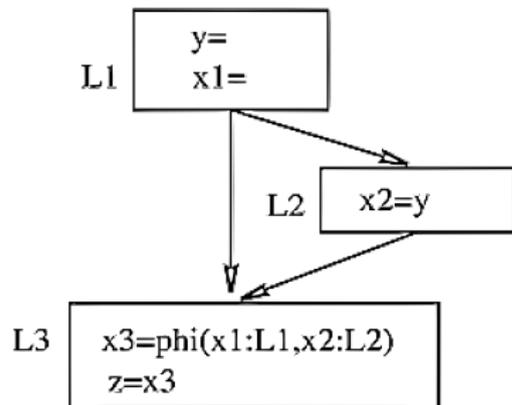
Rückwandlung
aus
SSA-Form

Rückwandlung aus SSA-Form 3



Jetzt Annahme: Einfache Optimierung hat stattgefunden

Vor Copy-Propagation



Nach Copy-Propagation

Rückwandlung durch einfaches Löschen ... geht schief:

- Phi-Funktion auflösen nach x oder y ?

OptComp

A. Koch

Organisatorisch

Einleitung

SSA
Erzeugen

SSA für
strukturierte
Sprachen

Dominatoren

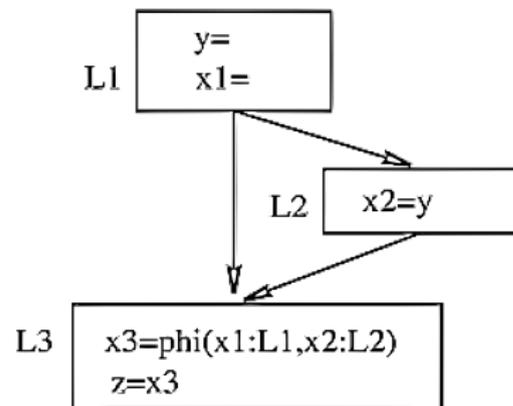
Rückwandlung
aus
SSA-Form

Rückwandlung aus SSA-Form 3

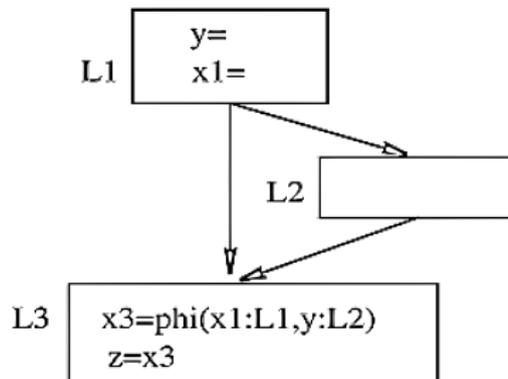


Jetzt Annahme: Einfache Optimierung hat stattgefunden

Vor Copy-Propagation



Nach Copy-Propagation



Rückwandlung durch einfaches Löschen ... geht schief:

- Phi-Funktion auflösen nach x oder y ?

OptComp

A. Koch

Organisatorisch

Einleitung

SSA
Erzeugen

SSA für
strukturierte
Sprachen

Dominatoren

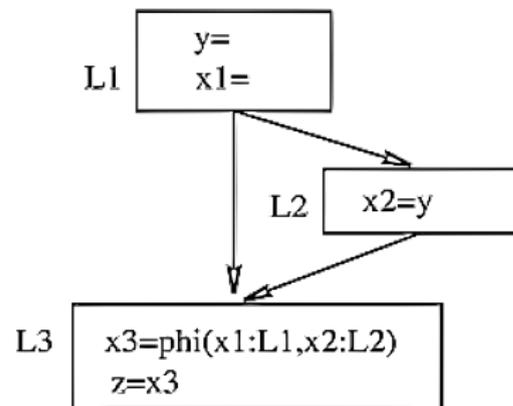
Rückwandlung
aus
SSA-Form

Rückwandlung aus SSA-Form 3

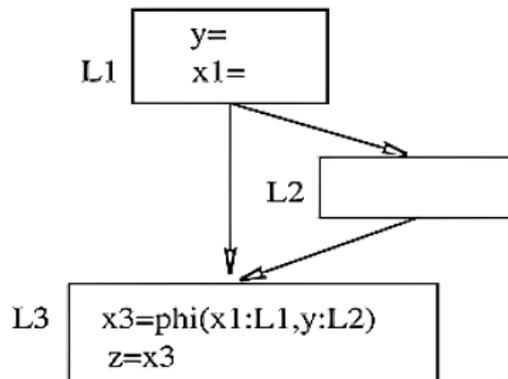


Jetzt Annahme: Einfache Optimierung hat stattgefunden

Vor Copy-Propagation



Nach Copy-Propagation



Rückwandlung durch einfaches Löschen ... geht schief:

- Phi-Funktion auflösen nach x oder y?

OptComp

A. Koch

Organisatorisch

Einleitung

SSA
Erzeugen

SSA für
strukturierte
Sprachen

Dominatoren

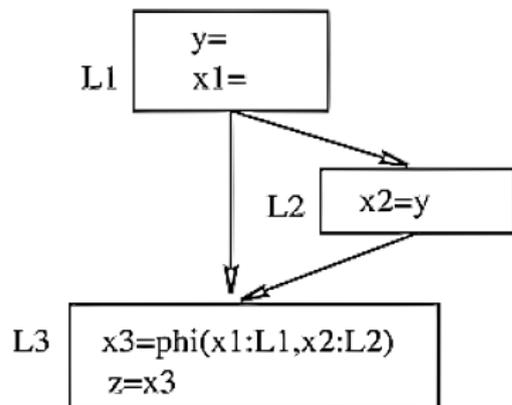
Rückwandlung
aus
SSA-Form

Rückwandlung aus SSA-Form 3

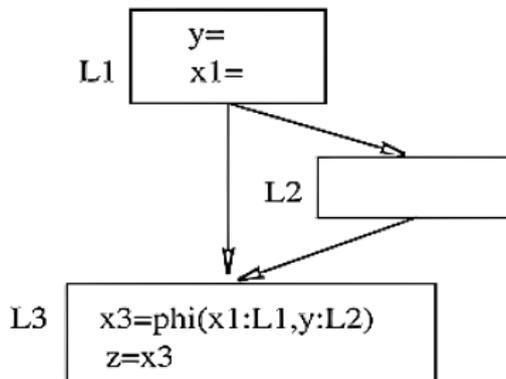


Jetzt Annahme: Einfache Optimierung hat stattgefunden

Vor Copy-Propagation



Nach Copy-Propagation



Rückwandlung durch einfaches Löschen ... geht schief:

- Phi-Funktion auflösen nach x oder y?

OptComp

A. Koch

Organisatorisch

Einleitung

SSA
Erzeugen

SSA für
strukturierte
Sprachen

Dominatoren

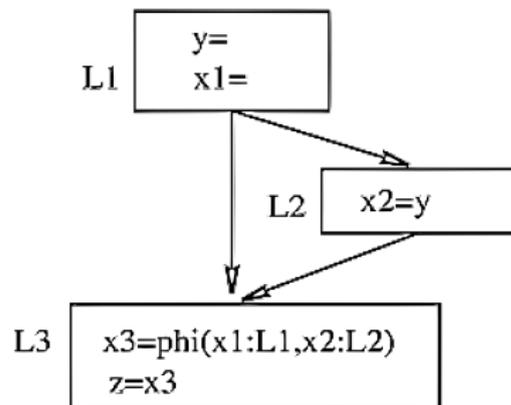
Rückwandlung
aus
SSA-Form

Rückwandlung aus SSA-Form 3

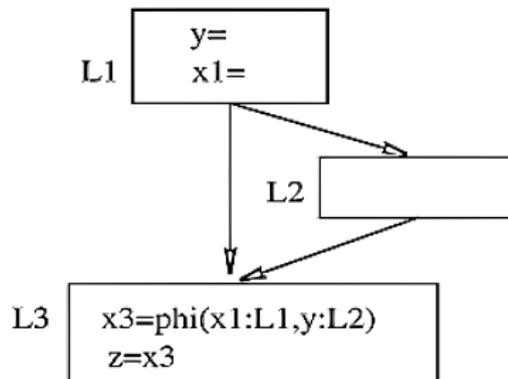


Jetzt Annahme: Einfache Optimierung hat stattgefunden

Vor Copy-Propagation



Nach Copy-Propagation



Rückwandlung durch einfaches Löschen ... geht schief:

- Phi-Funktion auflösen nach x oder y?

OptComp

A. Koch

Organisatorisch

Einleitung

SSA
Erzeugen

SSA für
strukturierte
Sprachen

Dominatoren

Rückwandlung
aus
SSA-Form

Rückwandlung aus SSA-Form 4



OptComp

A. Koch

Organisatorisch

Einleitung

SSA
Erzeugen

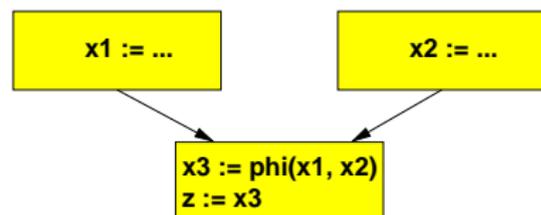
SSA für
strukturierte
Sprachen

Dominatoren

Rückwandlung
aus
SSA-Form

Besserer Ansatz: Füge Kopieroperationen in Vorgängerblöcke der Phi-Funktion ein

Vorher



Nachher

Zielführender als naives Löschen!

Rückwandlung aus SSA-Form 4



OptComp

A. Koch

Organisatorisch

Einleitung

SSA
Erzeugen

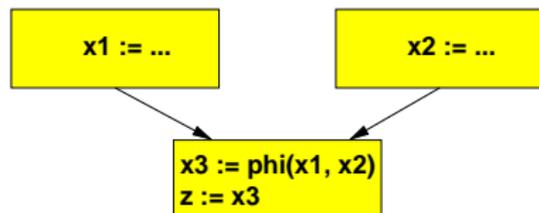
SSA für
strukturierte
Sprachen

Dominatoren

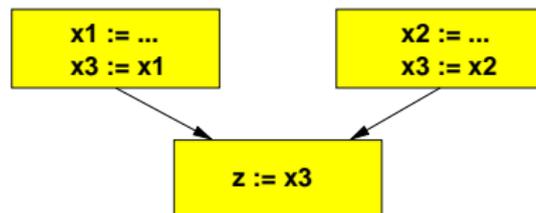
Rückwandlung
aus
SSA-Form

Besserer Ansatz: Füge Kopieroperationen in Vorgängerblöcke der Phi-Funktion ein

Vorher



Nachher



Zielführender als naives Löschen!

Rückwandlung aus SSA-Form 4



OptComp

A. Koch

Organisatorisch

Einleitung

SSA
Erzeugen

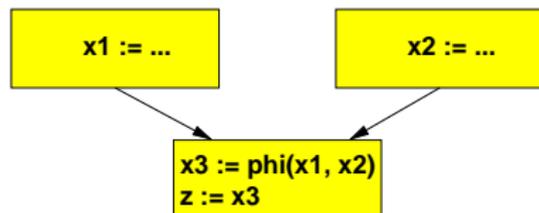
SSA für
strukturierte
Sprachen

Dominatoren

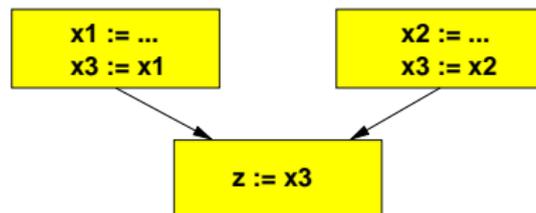
Rückwandlung
aus
SSA-Form

Besserer Ansatz: Füge Kopieroperationen in Vorgängerblöcke der Phi-Funktion ein

Vorher



Nachher



Zielführender als naives Löschen!

Rückwandlung aus SSA-Form 4



OptComp

A. Koch

Organisatorisch

Einleitung

SSA
Erzeugen

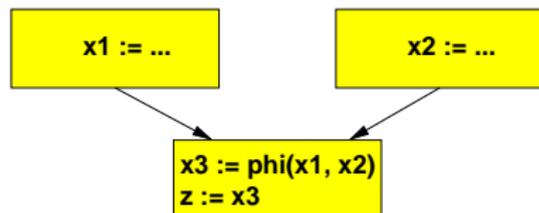
SSA für
strukturierte
Sprachen

Dominatoren

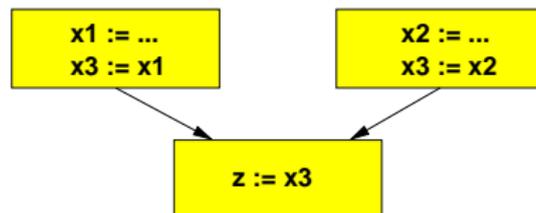
Rückwandlung
aus
SSA-Form

Besserer Ansatz: Füge Kopieroperationen in Vorgängerblöcke der Phi-Funktion ein

Vorher



Nachher



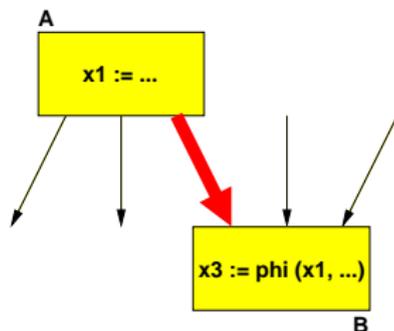
Zielführender als naives Löschen!



Problemfall: Kritische Kanten

Kritische Kontrollflusskante

Eine kritische Kante im CFG verläuft von einem Block mit mehreren Nachfolgern zu einem Block mit mehreren Vorgängern.

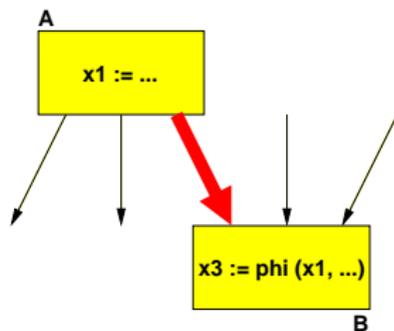


Rückwandlung aus der SSA-Form 6



Problem bei kritischen Kanten

- Wo Kopierzuweisungen von A bei Auflösen der Phi-Funktion in B unterbringen?
- Am Ende von A?
 - Geht nicht, da dann alle Nachfolger von A die Kopie für B bekommen!
- Am Anfang von B?
 - Geht nicht, da dann alle Vorgänger von B die Kopie von A bekommen!

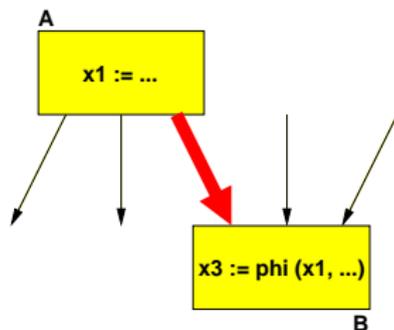


Rückwandlung aus der SSA-Form 6



Problem bei kritischen Kanten

- Wo Kopierzuweisungen von A bei Auflösen der Phi-Funktion in B unterbringen?
- Am Ende von A?
 - Geht nicht, da dann alle Nachfolger von A die Kopie für B bekommen!
- Am Anfang von B?
 - Geht nicht, da dann alle Vorgänger von B die Kopie von A bekommen!

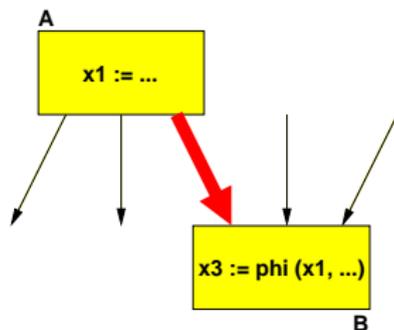


Rückwandlung aus der SSA-Form 6



Problem bei kritischen Kanten

- Wo Kopierzuweisungen von A bei Auflösen der Phi-Funktion in B unterbringen?
- Am Ende von A?
 - Geht nicht, da dann alle Nachfolger von A die Kopie für B bekommen!
- Am Anfang von B?
 - Geht nicht, da dann alle Vorgänger von B die Kopie von A bekommen!

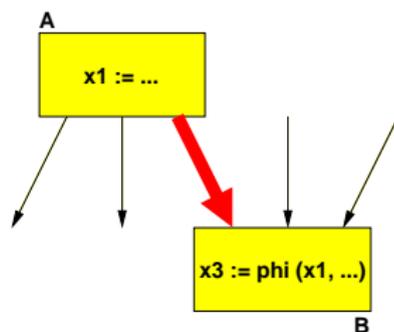


Rückwandlung aus der SSA-Form 6



Problem bei kritischen Kanten

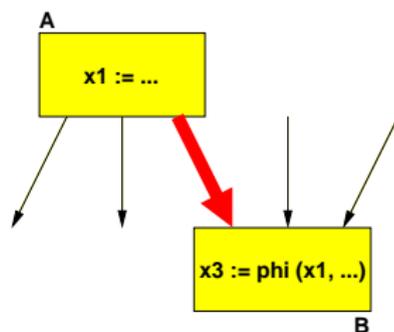
- Wo Kopierzuweisungen von A bei Auflösen der Phi-Funktion in B unterbringen?
- Am Ende von A?
 - Geht nicht, da dann alle Nachfolger von A die Kopie für B bekommen!
- Am Anfang von B?
 - Geht nicht, da dann alle Vorgänger von B die Kopie von A bekommen!





Problem bei kritischen Kanten

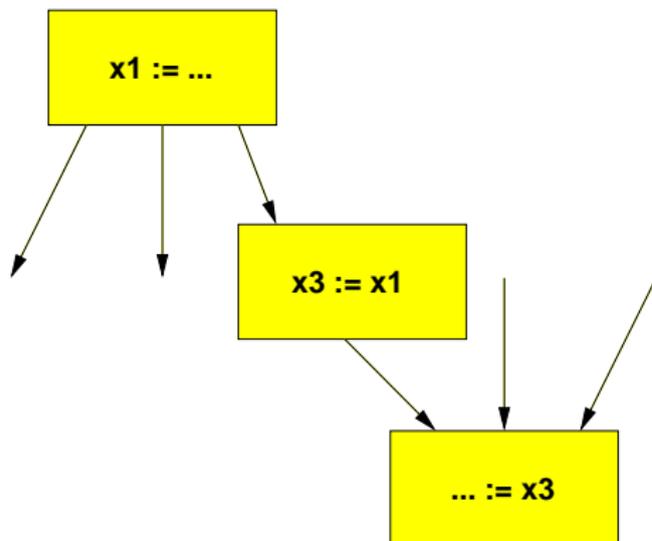
- Wo Kopierzuweisungen von A bei Auflösen der Phi-Funktion in B unterbringen?
- Am Ende von A?
 - Geht nicht, da dann alle Nachfolger von A die Kopie für B bekommen!
- Am Anfang von B?
 - Geht nicht, da dann alle Vorgänger von B die Kopie von A bekommen!



Rückwandlung aus der SSA-Form 7



Einfache Lösung:
Kante aufspalten und neuen Block einfügen!

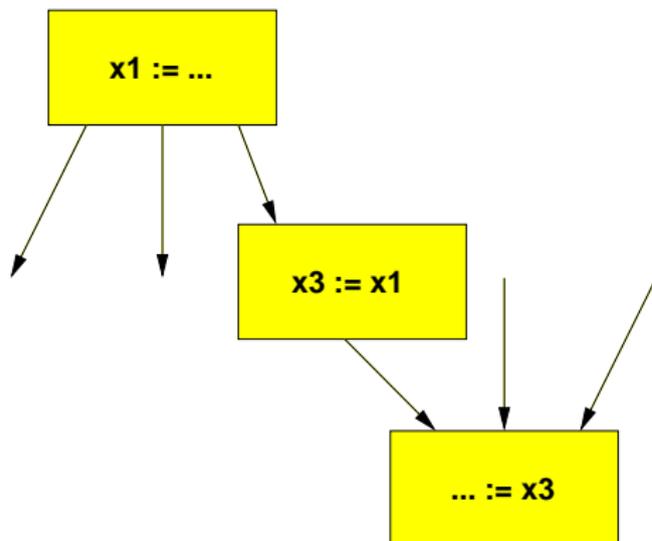


Funktioniert immer!

Rückwandlung aus der SSA-Form 7



Einfache Lösung:
Kante aufspalten und neuen Block einfügen!



Funktioniert immer!



Nachteil: Verlangsamt möglicherweise Programm

- Beispiel: Zusätzliche Sprunganweisung bei REPEAT/UNTIL

Abhilfe: Gezielteres Einfügen von Kopien

- Briggs 1998 oder Sreedhar 1999

Nicht ganz einfach, wir spalten kritische Kanten wenn nötig immer auf!



Nachteil: Verlangsamt möglicherweise Programm

- Beispiel: Zusätzliche Sprunganweisung bei REPEAT/UNTIL

Abhilfe: Gezielteres Einfügen von Kopien

- Briggs 1998 oder Sreedhar 1999

Nicht ganz einfach, wir spalten kritische Kanten wenn nötig immer auf!



Nachteil: Verlangsamt möglicherweise Programm

- Beispiel: Zusätzliche Sprunganweisung bei REPEAT/UNTIL

Abhilfe: Gezielteres Einfügen von Kopien

- Briggs 1998 oder Sreedhar 1999

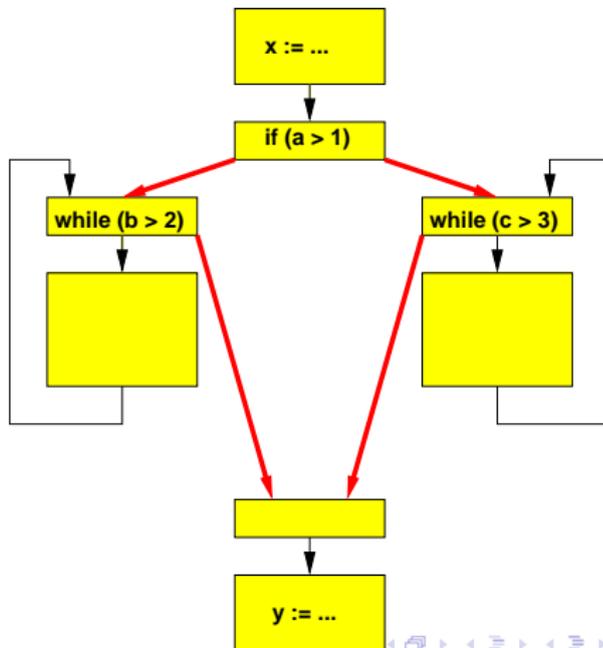
Nicht ganz einfach, wir spalten kritische Kanten wenn nötig immer auf!

Rückwandlung aus der SSA-Form 9



Praktisch: Können kritische Kanten in strukturierten CFGs à la Triangle auftreten?

```
x := ...  
if (a > 1) then {  
  while (b > 2) do {  
  }  
} else {  
  while (c > 3) do {  
  }  
}  
y := ...
```



OptComp

A. Koch

Organisatorisch

Einleitung

SSA
Erzeugen

SSA für
strukturierte
Sprachen

Dominatoren

Rückwandlung
aus
SSA-Form

Rückwandlung aus SSA-Form 10



- Aber nicht alle kritischen Kanten sind relevant
- Nur solche **vor** Blöcken mit phi-Funktionen

Damit Vorgehensweise zur Rückwandlung

- Teile phi-Funktion in Kopieranweisungen auf
- Lege Kopieranweisung am Ende des entsprechenden Vorgängerknotens ab
- **Es sei denn**, dass Kante zum Vorgänger kritisch ist
- **Dann** Kante aufspalten, Kopieranweisung in eingefügten Knoten legen

OptComp

A. Koch

Organisatorisch

Einleitung

SSA

Erzeugen

SSA für
strukturierte
Sprachen

Dominatoren

Rückwandlung
aus
SSA-Form

Rückwandlung aus SSA-Form 10



- Aber nicht alle kritischen Kanten sind relevant
- Nur solche **vor** Blöcken mit phi-Funktionen

Damit Vorgehensweise zur Rückwandlung

- Teile phi-Funktion in Kopieranweisungen auf
- Lege Kopieranweisung am Ende des entsprechenden Vorgängerknotens ab
- **Es sei denn**, dass Kante zum Vorgänger kritisch ist
- **Dann** Kante aufspalten, Kopieranweisung in eingefügten Knoten legen

OptComp

A. Koch

Organisatorisch

Einleitung

SSA

Erzeugen

SSA für
strukturierte
Sprachen

Dominatoren

Rückwandlung
aus
SSA-Form

Rückwandlung aus SSA-Form 10



- Aber nicht alle kritischen Kanten sind relevant
- Nur solche **vor** Blöcken mit phi-Funktionen

Damit Vorgehensweise zur Rückwandlung

- Teile phi-Funktion in Kopieranweisungen auf
- Lege Kopieranweisung am Ende des entsprechenden Vorgängerknotens ab
- **Es sei denn**, dass Kante zum Vorgänger kritisch ist
- **Dann** Kante aufspalten, Kopieranweisung in eingefügten Knoten legen

OptComp

A. Koch

Organisatorisch

Einleitung

SSA
Erzeugen

SSA für
strukturierte
Sprachen

Dominatoren

Rückwandlung
aus
SSA-Form



- Aber nicht alle kritischen Kanten sind relevant
- Nur solche **vor** Blöcken mit phi-Funktionen

Damit Vorgehensweise zur Rückwandlung

- Teile phi-Funktion in Kopieranweisungen auf
- Lege Kopieranweisung am Ende des entsprechenden Vorgängerknotens ab
- **Es sei denn**, dass Kante zum Vorgänger kritisch ist
- **Dann** Kante aufspalten, Kopieranweisung in eingefügten Knoten legen



- Aber nicht alle kritischen Kanten sind relevant
- Nur solche **vor** Blöcken mit phi-Funktionen

Damit Vorgehensweise zur Rückwandlung

- Teile phi-Funktion in Kopieranweisungen auf
- Lege Kopieranweisung am Ende des entsprechenden Vorgängerknotens ab
- **Es sei denn**, dass Kante zum Vorgänger kritisch ist
- **Dann** Kante aufspalten, Kopieranweisung in eingefügten Knoten legen



- Aber nicht alle kritischen Kanten sind relevant
- Nur solche **vor** Blöcken mit phi-Funktionen

Damit Vorgehensweise zur Rückwandlung

- Teile phi-Funktion in Kopieranweisungen auf
- Lege Kopieranweisung am Ende des entsprechenden Vorgängerknotens ab
- **Es sei denn**, dass Kante zum Vorgänger kritisch ist
- **Dann** Kante aufspalten, Kopieranweisung in eingefügten Knoten legen



- Aber nicht alle kritischen Kanten sind relevant
- Nur solche **vor** Blöcken mit phi-Funktionen

Damit Vorgehensweise zur Rückwandlung

- Teile phi-Funktion in Kopieranweisungen auf
- Lege Kopieranweisung am Ende des entsprechenden Vorgängerknotens ab
- **Es sei denn**, dass Kante zum Vorgänger kritisch ist
- **Dann** Kante aufspalten, Kopieranweisung in eingefügten Knoten legen



- Transformation in SSA-Form
 - Allgemeiner Fall (aus dem Orbit)
 - Sonderfall: Strukturierte Programmiersprachen
 - Berechnung von Dominatoren
 - Rückwandlung aus der SSA-Form

Nächste Aufgabe: Erzeuge aus AST CFG in SSA-Form
nach Methode von Brandis und Mössenböck



- Transformation in SSA-Form
- Allgemeiner Fall (aus dem Orbit)
- Sonderfall: Strukturierte Programmiersprachen
- Berechnung von Dominatoren
- Rückwandlung aus der SSA-Form

Nächste Aufgabe: Erzeuge aus AST CFG in SSA-Form
nach Methode von Brandis und Mössenböck



- Transformation in SSA-Form
- Allgemeiner Fall (aus dem Orbit)
- **Sonderfall: Strukturierte Programmiersprachen**
- Berechnung von Dominatoren
- Rückwandlung aus der SSA-Form

Nächste Aufgabe: Erzeuge aus AST CFG in SSA-Form
nach Methode von Brandis und Mössenböck



- Transformation in SSA-Form
- Allgemeiner Fall (aus dem Orbit)
- Sonderfall: Strukturierte Programmiersprachen
- Berechnung von Dominatoren
- Rückwandlung aus der SSA-Form

Nächste Aufgabe: Erzeuge aus AST CFG in SSA-Form nach Methode von Brandis und Mössenböck

OptComp

A. Koch

Organisatorisch

Einleitung

SSA
Erzeugen

SSA für
strukturierte
Sprachen

Dominatoren

Rückwandlung
aus
SSA-Form



- Transformation in SSA-Form
- Allgemeiner Fall (aus dem Orbit)
- Sonderfall: Strukturierte Programmiersprachen
- Berechnung von Dominatoren
- Rückwandlung aus der SSA-Form

Nächste Aufgabe: Erzeuge aus AST CFG in SSA-Form
nach Methode von Brandis und Mössenböck

OptComp

A. Koch

Organisatorisch

Einleitung

SSA
Erzeugen

SSA für
strukturierte
Sprachen

Dominatoren

Rückwandlung
aus
SSA-Form



- Transformation in SSA-Form
- Allgemeiner Fall (aus dem Orbit)
- Sonderfall: Strukturierte Programmiersprachen
- Berechnung von Dominatoren
- Rückwandlung aus der SSA-Form

Nächste Aufgabe: Erzeuge aus AST CFG in SSA-Form
nach Methode von Brandis und Mössenböck

OptComp

A. Koch

Organisatorisch

Einleitung

SSA
Erzeugen

SSA für
strukturierte
Sprachen

Dominatoren

Rückwandlung
aus
SSA-Form



- Transformation in SSA-Form
- Allgemeiner Fall (aus dem Orbit)
- Sonderfall: Strukturierte Programmiersprachen
- Berechnung von Dominatoren
- Rückwandlung aus der SSA-Form

Nächste Aufgabe: Erzeuge aus AST CFG in SSA-Form
nach Methode von Brandis und Mössenböck