

# Optimierende Compiler

Partielle Redundanzeliminierung (PRE)

#### Andreas Koch

FG Eingebettete Systeme und ihre Anwendungen Informatik, TU Darmstadt

Sommersemester 2010

oc

A. Koch

inleitung

Konzepte

Notation

E Dfodo

Dotonfluor





# Einleitung

OC

. Koch

Einleitung

. . . . . . . . . .

torizopio

Jotation

E-Pfade

tenfluss

## Common Subexpression Elimination



 Finde wiederholte Berechnungen auf einem Ausführungspfad

- ... und beseitige alle außer der ersten
- Bisher kennengelernte Verfahren
  - LVN, SVN, DVN
  - GCSE



A. Koch

Einleitung

Konzepte

E-Pfad PRI

Notation

E-Pfade





#### Totale Redundanz

Ein Ausdruck ist total redundant, wenn er auf allen Pfaden zu einer redundanten Verwendung berechnet wird.

#### Partielle Redundanz

Ein Ausdruck ist partiell redundant, wenn er auf einigen, aber nicht auf allen Pfaden zu einer redundanten Verwendung berechnet wird.

oc

A. Koch

Einleitung

Konzepte

E-Pfad PR

Notation

E-Pfade

Jatenfluss



#### Totale Redundanz

Ein Ausdruck ist total redundant, wenn er auf allen Pfaden zu einer redundanten Verwendung berechnet wird.

#### Partielle Redundanz

Ein Ausdruck ist partiell redundant, wenn er auf einigen, aber nicht auf allen Pfaden zu einer redundanten Verwendung berechnet wird.

OC

A. Koch

Einleitung
PRF

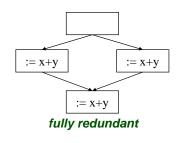
Konzepte

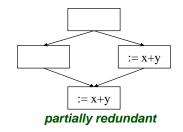
Notation

E-Pfade

Datenfluss



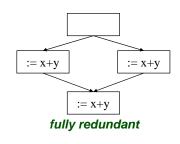


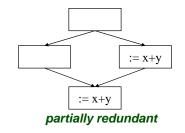


Einleitung









OC

A. Koch

Einleitung

1112

torizopie

Manatan

E-Pfade

atenfluss

CSE erkennt nur totale Redundanz.

Quelle: CMU CS 15-745 2009



# Bewegung von schleifeninvarianten Anweisungen



- Loop Invariant Code Motion
- Bewege Anweisungen, die jede Iteration denselben Wert liefern
  - Dafür müssen die Operanden schleifeninvariant sein
- ... aus der Schleife heraus

oc

A. Koch

Einleitung

Konzepte

Dfod DDI

Notation

-Pfade

Datenfluss

#### Schleifeninvarianz



Ein Operand ist schleifeninvariant, wenn

- er konstant ist, oder
- alle seine Definitionen außerhalb der Schleife liegen, oder
  - Erinnerung: Datenflußproblem REACHES "Erreichende Definitionen"
- er eine einzelne Definition innerhalb der Schleife hat, die aber selbst invariant ist.

OC

A. Kocn

Einleitung

Nonzepie

. . .

- 57 .



## **Loop Invariant Code Motion**



Naiver Ansatz: Bewege invariante Anweisungen **S** vor Schleifenkopf (pre-header).

Geht aber nur, wenn jedes S

- alle Verwendungen seiner LHS dominiert
- alle Schleifenausgänge dominiert (break, continue, ...)

UC

A. Koch

Einleitung

Konzepte

- D(- -I-

E-Flaue

Jaterniuss

## **Loop Invariant Code Motion**



Naiver Ansatz: Bewege invariante Anweisungen **S** vor Schleifenkopf (pre-header).

Geht aber nur, wenn jedes S

- alle Verwendungen seiner LHS dominiert
- alle Schleifenausgänge dominiert (break, continue, ...)

OC

A. Koch

Einleitung

/onzonto

rtorizopio

Notation

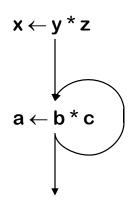
E-Pfade

atenfluss

## LICM und partielle Redundanz



Schleifeninvariante Ausdrücke sind eine Art der partiellen Redundanz



ОС

A. Koch

Einleitung

Konzepte

E-Pfad PR

Notation

E-Ptade

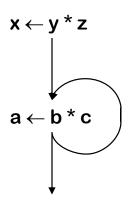
atenfluss

**b\*c** nur redundant auf Rückwärtskante.

## LICM und partielle Redundanz



Schleifeninvariante Ausdrücke sind eine Art der partiellen Redundanz



ОС

A. Kocn

Einleitung

Konzepte

:-Piad PRE

E-Pfade

atenfluss

**b**\*c nur redundant auf Rückwärtskante.



# Eliminierung partieller Redundanz

OC

A. Koch

RE

Konzepte

E-Pfad PR

Notation

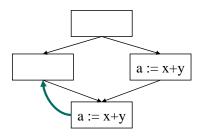
E-Pfade

atenfluss

## Eliminierung partieller Redundanz: Idee



- Bewege partiell redundante Berechnungen an ihre optimalen Stellen
  - Vermeide so Doppelberechnungen
- Beinhaltet CSE und LICM

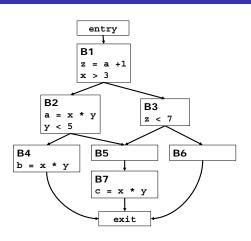


Einleitung
PRE
Konzepte
E-Pfad PRE
Notation
E-Pfade
Datenfluss

Quelle: CMU CS 15-745 2009







OC

A. NOCH

-interturing

RE

Konzepte

E-Pfad PR

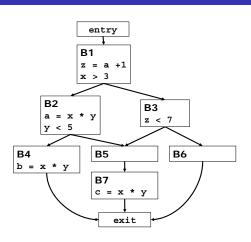
Notation

E-Pfade

atenfluss

Welche Ausdrücke sind partiell redundant?





A. NOCH

-interturing

RE

Conzepte

E-Pfad PRI

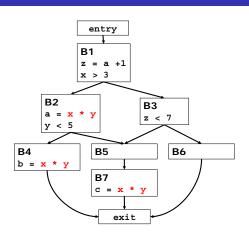
lotation

E-Pfade

atenfluss

Welche Ausdrücke sind partiell redundant?





A. NOCH

Intellung

KE

Konzepte

E-Pfad PR

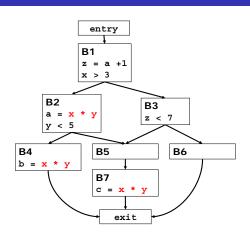
lotation

E-Pfade

Datenfluss

Wo die Berechnung tatsächlich durchführen, wo die Kopien verwenden?





A. Koch

A. NOCH

-interturing

RE

Conzepte

E-Piad PKI

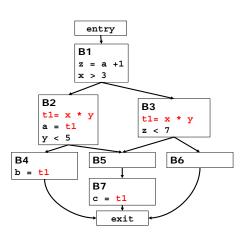
lotation

E-Pfade

atenfluss

Wo die Berechnung tatsächlich durchführen, wo die Kopien verwenden?





OC

A. Koch

inleitung

RE

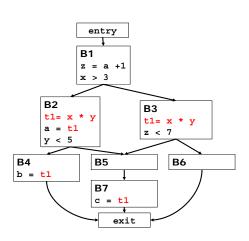
Konzepte

- 54

- 1 1000

Datenfluss





OC

A. Koch

inleitung

KE

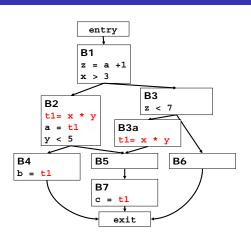
Konzepte

- - - -

E-Flaue

atenfluss





A. NOCH

-interturing

KE

Konzepte

E-Pfad PR

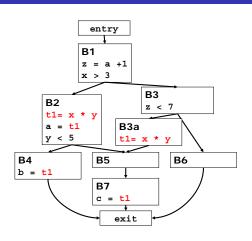
votation

E-Pfade

atenfluss

Was ist besonders an Kante (B3, B5)?





OC

A. NOCH

inleitung

RE

Konzepte

E-Pfad PR

otation

E-Ptade

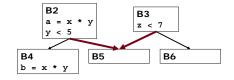
atenfluss

Was ist besonders an Kante (B3, B5)?

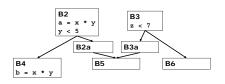
#### Kritische Kante



#### Vor Aufteilen



#### Alle kritischen Kanten aufgeteilt



Nachteil: Potentiell langsamer (Compile- und Laufzeit!)



OC A. Koch

inleitung

'RE

Konzepte

:-riau rni . . . ..

Pfado

-Pfade

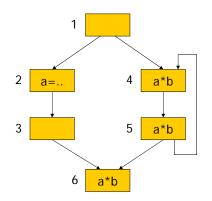
## PRE kombiniert drei Optimierungen



• CSE: a\*b in **B5** 

LICM: a\*b in B4

Bewegung: a\*b von B6 nach B3



ОС

A. NOCH

.iriieituriy

KE

Conzepte

ototion

E-Pfade

atenfluss

Quelle: Dhamdhere, Code optimization by partial redundancy elimination using Eliminatability paths (E-paths)



#### Qualitätskriterien



#### Optimale Lebenszeit

Die Lebenszeit von einer Neuberechnung zu einer Verwendung sollte so kurz wie möglich sein.

⇒Benötigt weniger Register

#### Berechnungsoptimalität

Zur Programmlaufzeit sollen so wenige Berechnungen wie möglich ausgeführt werden.

OC

A. Koch

PRF

onzente

\_ Dto4 DD

Notation

E-Pfade

Datenfluss

#### Qualitätskriterien



#### Optimale Lebenszeit

Die Lebenszeit von einer Neuberechnung zu einer Verwendung sollte so kurz wie möglich sein.

⇒Benötigt weniger Register

#### Berechnungsoptimalität

Zur Programmlaufzeit sollen so wenige Berechnungen wie möglich ausgeführt werden.

OC

A. KOCII

PRF

onzepte

E-Pfad PR

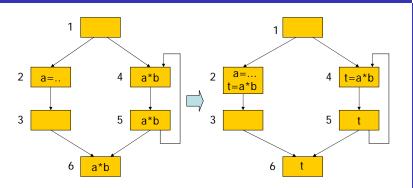
Notation

z-Pfade

Datenfluss

## Algorithmus von Morel und Renvoise 1979





- Neue Berechnung von a\*b eingefügt in B2
- B3 wäre besser (kürzere Lebenszeit!)
- a\*b in B4 nicht berechnungsoptimal
  - Wegen (B1,B2) nicht am Ende von B1 eingefügt
- a\*b gesichert in B2+B4, wiederverwendet in B5+B6

OC

4. Noci

inieitung

PRE

vousebre

E-Plau PRI

- Dr.

E-Pfade

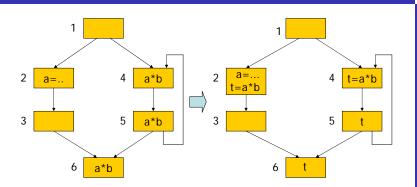
atenfluss

#### Algorithmus von Morel und Renvoise 1979



A Koch

**PRF** 

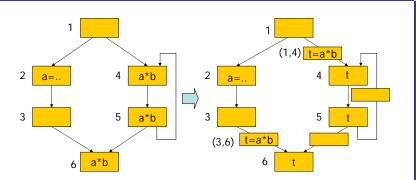


- Neue Berechnung von a\*b eingefügt in B2
- B3 wäre besser (kürzere Lebenszeit!)
- a\*b in B4 nicht berechnungsoptimal
  - Wegen (B1,B2) nicht am Ende von B1 eingefügt
- a\*b gesichert in B2+B4, wiederverwendet in B5+B6



# Alg. von Knoop, Ruthing und Steffen 1992 Lazy Code Motion (LCM)





- Alle Kanten zu Join-Knoten aufteilen
- a\*b einfügen in (B3,B6) (hat optimale Lebenszeit!)
- a\*b einfügen in (B1,B4) (berechnungsoptimal)
- Erzeugt zusätzliche Blöcke (leere entfernbar)

Einleitung
PRE
Konzepte

A Koch

Notation

E-Pfade

atenfluss

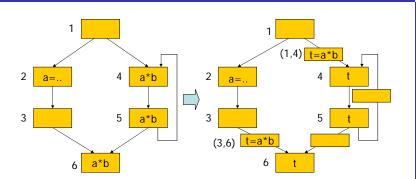


#### Alg. von Knoop, Ruthing und Steffen 1992 Lazy Code Motion (LCM)



A Koch

PRF



- Alle Kanten zu Join-Knoten aufteilen
- a\*b einfügen in (B3,B6) (hat optimale Lebenszeit!)
- a\*b einfügen in (B1,B4) (berechnungsoptimal)
- Erzeugt zusätzliche Blöcke (leere entfernbar)



# Konzepte

OC

I. Koch

nleitung

Konzepte

onzepte

dotation

E-Pfade

tenfluss

## Verfügbarkeit



#### Verfügbarkeit (Availability)

Ein Ausdruck *e* ist an einer Programmstelle *p* verfügbar, wenn sein Wert auf allen Pfaden von Programmanfang zu *p* berechnet wird.

→Totale Redundanz von e an p

#### Partielle Verfügbarkeit (Partial Availability

Ein Ausdruck *e* ist an einer Programmstelle *p* partiell verfügbar, wenn sein Wert auf einigen Pfaden von Programmanfang zu *p* berechnet wird.

⇒Partielle Redundanz von e an p

oc

A. Kocn

ileiturig

Konzepte

L-i iau i ii

Diada

Datonfluce

4 D > 4 D > 4 E > 4 E > E 990

## Verfügbarkeit



#### Verfügbarkeit (Availability)

Ein Ausdruck *e* ist an einer Programmstelle *p* verfügbar, wenn sein Wert auf allen Pfaden von Programmanfang zu *p* berechnet wird.

➡Totale Redundanz von e an p

#### Partielle Verfügbarkeit (Partial Availability)

Ein Ausdruck *e* ist an einer Programmstelle *p* partiell verfügbar, wenn sein Wert auf einigen Pfaden von Programmanfang zu *p* berechnet wird.

⇒Partielle Redundanz von e an p

oc

A. Koch

nieitung RE

Konzepte

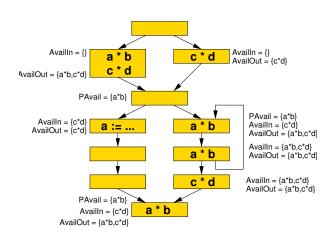
\_-r lau r ni

E-Pfade

Datenfluss

## Beispiel: Verfügbare Ausdrücke





OC

A. Koch

inieitung

Konzepte

F-Pfad PF

Notation

E-Pfade

atenfluss

#### Lokal exponierte Ausdrücke



Nach oben exponierte Ausdrücke (Upwards Exposed)

Ausdrücke, deren Operanden vom Blockanfang bis zu ihrer Stelle nicht überschrieben werden sind nach oben exponiert.

- ⇒Ihre Berechnung könnte an den Blockanfang vorgezogen werden
  - ullet o UEExpr
  - Auch genannt lokal vorziehbar (locally anticipatable, ANTloc)

OC

A. Koch

nleitung

Konzepte

E-PIAU PRE

-Pfade

atenfluss

### Lokal exponierte Ausdrücke



### Nach oben exponierte Ausdrücke (Upwards Exposed)

Ausdrücke, deren Operanden vom Blockanfang bis zu ihrer Stelle nicht überschrieben werden sind nach oben exponiert.

- ⇒Ihre Berechnung könnte an den Blockanfang vorgezogen werden
  - ullet  $\rightarrow$  UEExpr
  - Auch genannt lokal vorziehbar (locally anticipatable, ANTloc)

OC

A. Koch

nleitung

Konzepte

:-Piad PRE

E-Pfade

### Lokal exponierte Ausdrücke



Nach unten exponierte Ausdrücke (Downward Exposed)

Ausdrücke, deren Operanden von ihrer Stelle bis zum Blockende nicht überschrieben werden sind nach unten exponiert.

⇒Ihre Berechnung könnte an das Blockende verzögert werden

ullet ightarrow DEExpr

OC

A. Koch

inleitung

Konzepte

-Pfad PRE

lotation

E-Pfade

### Lokal exponierte Ausdrücke



Nach unten exponierte Ausdrücke (Downward Exposed)

Ausdrücke, deren Operanden von ihrer Stelle bis zum Blockende nicht überschrieben werden sind nach unten exponiert.

- ⇒Ihre Berechnung könnte an das Blockende verzögert werden
  - ullet  $\rightarrow$  DEExpr

oc

A. Koch

nleitung

Konzepte

-Pfad PRE

Notation

E-Pfade

## Beispiel: Lokal exponierte Ausdrücke



a:=42

w := a + b

x := c + d

y := e + f

z := a + e

e := 23

nicht UEExpr/ANTloc, DEExpr

UEExpr/ANTloc, DEExpr

UEExpr/ANTloc, nicht DEExpr

nicht UEExpr/ANTloc, nicht DEExpr

OC

A. Koch

inleitung

Konzepte

lotation

E-Pfade

### Globale Vorziehbarkeit



Konzepte

### Vorziehbarkeit (Anticipatability)

Die Berechnung eines Ausdrucks e ist vorziehbar an eine Programmstelle p, wenn er auf allen Pfaden von p zum Programmende berechnet wird.

→ Auch genannt: Very Busy Expression

## Beispiel: Vorziehbarkeit





- a\*b vorziehbar nach Block 12
- Aber nicht nach Block 22

Konzepte

Quelle: Dhamdhere. Code optimization by partial redundancy elimination using Eliminatability paths (E-paths)



### Beispiel: Vorziehbarkeit





- a\*b vorziehbar nach Block 12
- Aber nicht nach Block 22

Datenfluss

Konzepte

Quelle: Dhamdhere, Code optimization by partial redundancy elimination using Eliminatability paths (E-paths)





### Sichere Berechnung

Ein Ausdruck *e* kann an der Stelle *p* sicher berechnet werden, wenn er dort bereits verfügbar ist oder dorthin vorgezogen werden kann.

⇒Ziel: Gleicher Wert ohne weitere eventuelle Berechnungsfehler (exceptions, z.B. Division-by-Zero)

- Im 1. Fall wurde der Ausdruck so bereits berechnet und könnte gefahrlos noch ein weiteres Mal berechnet werden
- Im 2. Fall würden eventuelle Fehler ohnehin auftreten, da dieser Ausdruck später in jedem Fall berechnet würde

oc

A. Koch

illeiturig

Konzepte

E-Pfad PRI

Notation

E-Pfade

ロ ト 4 個 ト 4 差 ト 4 差 ト 9 Q (C)



### Sichere Berechnung

Ein Ausdruck *e* kann an der Stelle *p* sicher berechnet werden, wenn er dort bereits verfügbar ist oder dorthin vorgezogen werden kann.

⇒Ziel: Gleicher Wert ohne weitere eventuelle Berechnungsfehler (exceptions, z.B. Division-by-Zero)

 Im 1. Fall wurde der Ausdruck so bereits berechnet und könnte gefahrlos noch ein weiteres Mal berechnet werden

 Im 2. Fall würden eventuelle Fehler ohnehin auftreten, da dieser Ausdruck später in jedem Fall berechnet würde OC

A. Koch

nieitung

Konzepte

E-Pfad PRI

Notation

E-Pfade



### Sichere Berechnung

Ein Ausdruck *e* kann an der Stelle *p* sicher berechnet werden, wenn er dort bereits verfügbar ist oder dorthin vorgezogen werden kann.

⇒Ziel: Gleicher Wert ohne weitere eventuelle Berechnungsfehler (exceptions, z.B. Division-by-Zero)

- Im 1. Fall wurde der Ausdruck so bereits berechnet und könnte gefahrlos noch ein weiteres Mal berechnet werden
- Im 2. Fall würden eventuelle Fehler ohnehin auftreten, da dieser Ausdruck später in jedem Fall berechnet würde

OC

A. Koch

oe

Konzepte

E-Pfad PRE

otation



#### Sichere Berechnung

Ein Ausdruck *e* kann an der Stelle *p* sicher berechnet werden, wenn er dort bereits verfügbar ist oder dorthin vorgezogen werden kann.

⇒Ziel: Gleicher Wert ohne weitere eventuelle Berechnungsfehler (exceptions, z.B. Division-by-Zero)

- Im 1. Fall wurde der Ausdruck so bereits berechnet und könnte gefahrlos noch ein weiteres Mal berechnet werden
- Im 2. Fall würden eventuelle Fehler ohnehin auftreten, da dieser Ausdruck später in jedem Fall berechnet würde

OC

A. Koch

inieitung RE

Konzepte

-Piau PRE

-Pfade

### Beispiel: Sicherheit

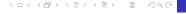




- Neuberechnung von a\*b in Block 12 ist sicher
- a\*b in Kante (Block 22, Block 23) wäre sicher

Konzepte

Quelle: Dhamdhere ,Code optimization by partial redundancy elimination using Eliminatability paths (E-paths)



### Beispiel: Sicherheit





- Neuberechnung von a\*b in Block 12 ist sicher
- a\*b in Block 22 ist unsicher
- a\*b in Kante (Block 22, Block 23) wäre sicher

Konzepte

Quelle: Dhamdhere ,Code optimization by partial redundancy elimination using Eliminatability paths (E-paths)



### Beispiel: Sicherheit





- Neuberechnung von a\*b in Block 12 ist sicher
- a\*b in Block 22 ist unsicher.
- a\*b in Kante (Block 22, Block 23) wäre sicher

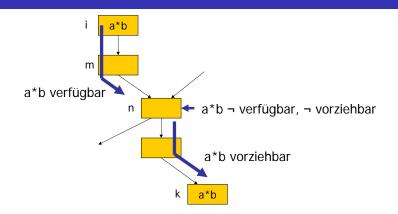
Konzepte

Quelle: Dhamdhere ,Code optimization by partial redundancy elimination using Eliminatability paths (E-paths)



### Beispiel: Verfügbar, vorziehbar, sicher







A. Koch

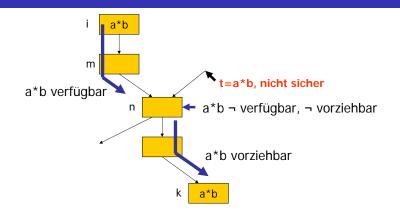
inleitung

Konzepte

-Pfade

### Beispiel: Verfügbar, vorziehbar, sicher







A. Kocn

inieitung

Konzepte

E-Pfad PR

lotation

E-Pfade

### Transparenz und Leere



#### Transparenz

Ein Block **B** ist transparent in Hinblick auf einen Ausdruck *e*, wenn er selbst keine Zuweisungen an Operanden von *e* enthält.

#### Leere

Ein Block **B** ist leer in Hinblick auf einen Ausdruck *e*, wenn er selbst weder eine Auswertung von *e* enthält noch Zuweisungen an Operanden von *e*.

⇒Schreibweise: empty(B) = TRUE bezüglich e

UC

A. Koch

nleitung

'KE

Konzepte

\_ I IAU I IIL

- 54 1

### Transparenz und Leere



### Transparenz

Ein Block **B** ist transparent in Hinblick auf einen Ausdruck *e*, wenn er selbst keine Zuweisungen an Operanden von *e* enthält.

#### Leere

Ein Block **B** ist leer in Hinblick auf einen Ausdruck *e*, wenn er selbst weder eine Auswertung von *e* enthält noch Zuweisungen an Operanden von *e*.

⇒Schreibweise: empty(**B**) = TRUE bezüglich *e* 

oc

A. Koch

inieitung

Konzepte

E-Pfad PRE

Notation

Detenflues

### Transparenz und Leere



### Transparenz

Ein Block **B** ist transparent in Hinblick auf einen Ausdruck *e*, wenn er selbst keine Zuweisungen an Operanden von *e* enthält.

#### Leere

Ein Block **B** ist leer in Hinblick auf einen Ausdruck *e*, wenn er selbst weder eine Auswertung von *e* enthält noch Zuweisungen an Operanden von *e*.

⇒Schreibweise: empty(**B**) = TRUE bezüglich *e* 

OC

A. Koch

leitung

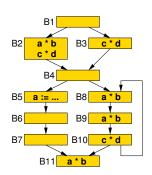
'KE

Konzepte

=-PIAU PRE

E-Pfade





#### • Für a\*b

- Alle Blöcke ausser B5 sind transparent
- b ∈ { B1, B3, B4, B6, B7, B10 }: empty(b) = TRUE

#### • Für c\*d

- Alle Blöcke sind transparent
- b ∈ { B1, B4, B5, B6,
   B7, B8, B9, B11 }:
   emptv(b) = TRUE



A. Koch

nleitung

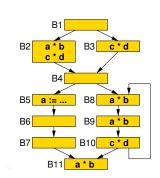
Konzepte

. F₋Pfad PRI

Notation

z-Piade





#### Für a\*b

- Alle Blöcke ausser B5 sind transparent
- b ∈ { B1, B3, B4, B6, B7, B10 }:empty(b) = TRUE
- Für c\*d
  - Alle Blöcke sind transparent
  - b ∈ { B1, B4, B5, B6, B7, B8, B9, B11 }:
     empty(b) = TRUF



A. Koch

inleitung

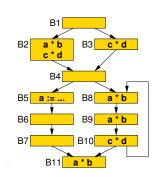
Konzepte

E-Pfad PRE

Notation

--Piaue





#### Für a\*b

- Alle Blöcke ausser B5 sind transparent
- b ∈ { B1, B3, B4, B6, B7, B10 }:
   empty(b) = TRUE
- Für c\*d
  - Alle Blöcke sind transparent
  - b ∈ { B1, B4, B5, B6, B7, B8, B9, B11 }:
     emotv(b) = TRUF



A. Koch

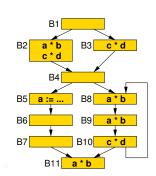
nleitung

Konzepte

E-Pfad PRE

E-Pfade





#### Für a\*b

- Alle Blöcke ausser B5 sind transparent
- b ∈ { B1, B3, B4, B6, B7, B10 }: empty(b) = TRUE
- Für c\*d
  - Alle Blöcke sind transparent
  - b ∈ { B1, B4, B5, B6, B7, B8, B9, B11 }: empty(b) = TRUE



A. Koch

nleitung

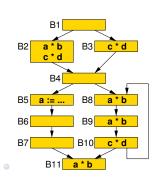
PRE

Konzepte

Notation

E-Pfade





Für a\*b

- Alle Blöcke ausser B5 sind transparent
- b ∈ { B1, B3, B4, B6, B7, B10 }:
   empty(b) = TRUE
- Für c\*d
  - Alle Blöcke sind transparent
  - b ∈ { B1, B4, B5, B6, B7, B8, B9, B11 }: empty(b) = TRUE

ОС

A. Koch

inleitung

Konzepte

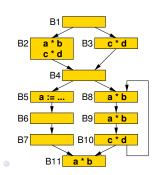
E-Pfad PRI

lotation

:-Ptade

4 D > 4 B > 4 B > 4 B > 9 Q P





Für a\*b

- Alle Blöcke ausser B5 sind transparent
- b ∈ { B1, B3, B4, B6, B7, B10 }: empty(b) = TRUE
- Für c\*d
  - Alle Blöcke sind transparent
  - b ∈ { B1, B4, B5, B6, B7, B8, B9, B11 }: empty(b) = TRUE



A. Koch

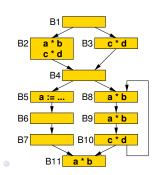
inleitung

Konzepte

E-Piad PKE

E-Pfade





Für a\*b

- Alle Blöcke ausser B5 sind transparent
- b ∈ { B1, B3, B4, B6, B7, B10 }: empty(b) = TRUE
- Für c\*d
  - Alle Blöcke sind transparent
  - b ∈ { B1, B4, B5, B6, B7, B8, B9, B11 }: empty(b) = TRUE



A. Koch

inleitung

Konzepte

E-Piad PKE

E-Pfade



- Bearbeite alle partiell redundanten Verwendungen von Ausdrücken e
  - Berechnungsoptimalität
- Füge Neuberechnungen von e an sicheren Stellen ein
- Lösche nun total redundant gewordene Berechnungen von e
- Achte auf kurze Lebenszeiten von neuberechneten Werten
- Vermeide unnötiges Aufteilen von Kanten
- Bevorzuge schnelleren Algorithmus
- Am besten auch noch möglichst einfach zu verstehen

ОС

A. Koch

inleitung

Konzepte

Notation

E-Pfade



- Bearbeite alle partiell redundanten Verwendungen von Ausdrücken e
  - Berechnungsoptimalität
- Füge Neuberechnungen von e an sicheren Stellen ein
- Lösche nun total redundant gewordene Berechnungen von e
- Achte auf kurze Lebenszeiten von neuberechneten Werten
- Vermeide unnötiges Aufteilen von Kanten
- Bevorzuge schnelleren Algorithmus
- Am besten auch noch möglichst einfach zu verstehen

ОС

A. Koch

inleitung

Konzepte

E-Piad PRE

E-Pfade



- Bearbeite alle partiell redundanten Verwendungen von Ausdrücken e
  - Berechnungsoptimalität
- Füge Neuberechnungen von e an sicheren Stellen ein
- Lösche nun total redundant gewordene Berechnungen von e
- Achte auf kurze Lebenszeiten von neuberechneten Werten
- Vermeide unnötiges Aufteilen von Kanten
- Bevorzuge schnelleren Algorithmus
- Am besten auch noch möglichst einfach zu verstehen

OC

A. Koch

inleitung

Konzepte

E-Pfad PRE

=\_Pfade



- Bearbeite alle partiell redundanten Verwendungen von Ausdrücken e
  - Berechnungsoptimalität
- Füge Neuberechnungen von e an sicheren Stellen ein
- Lösche nun total redundant gewordene Berechnungen von e
- Achte auf kurze Lebenszeiten von neuberechneten Werten
- Vermeide unnötiges Aufteilen von Kanten
- Bevorzuge schnelleren Algorithmus
- Am besten auch noch möglichst einfach zu verstehen



A. Koch

nleitung

Konzepte

. . . .

E-Pfade

Daterilluss





- Bearbeite alle partiell redundanten Verwendungen von Ausdrücken e
  - Berechnungsoptimalität
- Füge Neuberechnungen von e an sicheren Stellen ein
- Lösche nun total redundant gewordene Berechnungen von e
- Achte auf kurze Lebenszeiten von neuberechneten Werten
- Vermeide unnötiges Aufteilen von Kanten
- Bevorzuge schnelleren Algorithmus
- Am besten auch noch möglichst einfach zu verstehen



A. Koch

inleitung

Konzepte

-PIAU PRE

-Pfade



- Bearbeite alle partiell redundanten Verwendungen von Ausdrücken e
  - Berechnungsoptimalität
- Füge Neuberechnungen von e an sicheren Stellen ein
- Lösche nun total redundant gewordene Berechnungen von e
- Achte auf kurze Lebenszeiten von neuberechneten Werten
- Vermeide unnötiges Aufteilen von Kanten
- Bevorzuge schnelleren Algorithmus
- Am besten auch noch möglichst einfach zu verstehen



A. Koch

inleitung RE

Konzepte

-Plau PRE

E-Pfade



- Bearbeite alle partiell redundanten Verwendungen von Ausdrücken e
  - Berechnungsoptimalität
- Füge Neuberechnungen von e an sicheren Stellen ein
- Lösche nun total redundant gewordene Berechnungen von e
- Achte auf kurze Lebenszeiten von neuberechneten Werten
- Vermeide unnötiges Aufteilen von Kanten
- Bevorzuge schnelleren Algorithmus
- Am besten auch noch möglichst einfach zu verstehen

OC

A. Koch

inleitung

Konzepte

-Pfad PRE

lotation

=-Pfade



- Bearbeite alle partiell redundanten Verwendungen von Ausdrücken e
  - Berechnungsoptimalität
- Füge Neuberechnungen von e an sicheren Stellen ein
- Lösche nun total redundant gewordene Berechnungen von e
- Achte auf kurze Lebenszeiten von neuberechneten Werten
- Vermeide unnötiges Aufteilen von Kanten
- Bevorzuge schnelleren Algorithmus
- Am besten auch noch möglichst einfach zu verstehen

OC

A. Koch

inleitung RE

Konzepte

z-Piad PRE

E-Pfade

atennuss



# PRE mit Eliminierungspfaden

oc

A. Koch

RE

Konzepte

E-Pfad PRE

Notation

E-Pfade

### PRE mit Eliminierungspfaden



E-path\_PRE – Partial Redundancy Elimination Made Easy ACM SIGPLAN Notices, 2002, vol. 37, no 8, pp. 53-65

- Dhanajay M. Dhamdhere
- Korrigierte Fassung auf OC Web-Site
- Verfeinert zusammen mit Dheeraj Kumar 2006

OC

A. Koch

nleitung RE

Konzepte

F-Pfad PRF

Notation

E-Pfade

# PRE mit Eliminierungspfaden



E-path\_PRE – Partial Redundancy Elimination Made Easy ACM SIGPLAN Notices, 2002, vol. 37, no 8, pp. 53-65

- Dhanajay M. Dhamdhere
- Korrigierte Fassung auf OC Web-Site
- Verfeinert zusammen mit Dheeraj Kumar 2006

OC

A. NOCH

RE

F-Pfad PRF

Motation

F-Pfade

# PRE mit Eliminierungspfaden



E-path\_PRE – Partial Redundancy Elimination Made Easy ACM SIGPLAN Notices, 2002, vol. 37, no 8, pp. 53-65

- Dhanajay M. Dhamdhere
- Korrigierte Fassung auf OC Web-Site
- Verfeinert zusammen mit Dheeraj Kumar 2006

OC

A. Koch

RE

F-Pfad PRF

Notation



# Eliminierbarkeitspfad (E-Pfad)



#### Eliminierbarkeitspfad (E-Pfad, eliminatability path)

Ein E-Pfad für einen Ausdruck e ist ein Pfad  $b_i, \ldots, b_k$  im CFG so dass

- lacktriangle e lokal verfügbar in  $b_i$  und lokal vorziehbar in  $b_k$  ist
- ② Für  $b \in (b_i, ..., b_k)$  gilt: empty(b) = TRUE
- **3** *e* ist sicher auf jeder Ausgangskante eines Blocks  $b \in [b_i, \dots, b_k)$

Notation: Ein Pfad  $[b_i, \ldots, b_k]$  enthält seine Anfangs- und Endblöcke, ein Pfad  $(b_i, \ldots, b_k)$  nicht.

UC

A. Kocn

ileitung

·----

F-Pfad PRF

Notation

E-Pfade

# Eliminierbarkeitspfad (E-Pfad)



#### Eliminierbarkeitspfad (E-Pfad, eliminatability path)

Ein E-Pfad für einen Ausdruck e ist ein Pfad  $b_i, \ldots, b_k$  im CFG so dass

- lokal verfügbar in  $b_i$  und lokal vorziehbar in  $b_k$  ist
- ② Für  $b \in (b_i, ..., b_k)$  gilt: empty(b) = TRUE
- **3** *e* ist sicher auf jeder Ausgangskante eines Blocks  $b \in [b_i, \dots, b_k)$

Notation: Ein Pfad  $[b_i, \ldots, b_k]$  enthält seine Anfangs- und Endblöcke, ein Pfad  $(b_i, \ldots, b_k)$  nicht.

UC

A. NOCH

nleitung

Konzepte

E-Pfad PRE

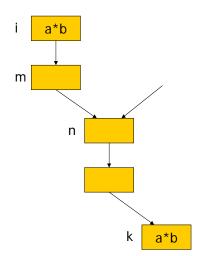
D(- -|-

\_-1 laut

# Beispiel: E-Pfad



- **a\*b** ist verfügbar an Ende von  $[i \dots m]$
- a\*b ist vorziehbar am
   Ende von [n...k)
- Berechnung von a\*b in Block k ist eliminierbar



OC

A. NOCH

PRF

Konzepte

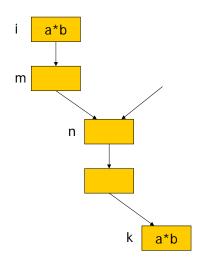
E-Pfad PRE

E-Pfade

# Beispiel: E-Pfad



- **a\*b** ist verfügbar an Ende von [i...m]
- a\*b ist vorziehbar am Ende von [n...k)
- Berechnung von a\*b in Block k ist eliminierbar



OC

A. Kocn

=inieitung >RF

Konzepte

F-Pfad PRF

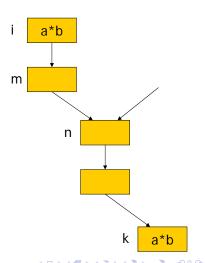
Notation

E-Pfade

### Beispiel: E-Pfad



- **a\*b** ist verfügbar an Ende von [i...m]
- a\*b ist vorziehbar am Ende von [n...k)
- Berechnung von a\*b in Block k ist eliminierbar



OC

A. KOCII

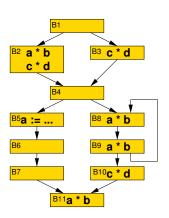
Einleitung PRE

Konzepte

E-Pfad PRE

E-Pfade





Für a\*b

- $[b_8, b_9]$
- $[b_9, b_8]$
- $\bullet$  [ $b_9$ ,  $b_{10}$ ,  $b_{11}$ ]
- Nicht:  $[b_2, b_4, b_8]$
- Für c\*d
  - $\bullet$  [ $b_2$ ,  $b_4$ ,  $b_8$ ,  $b_9$ ,  $b_{10}$ ]
  - $\bullet$  [ $b_3$ ,  $b_4$ ,  $b_8$ ,  $b_9$ ,  $b_{10}$

OC

A. Kocn

Einleitung

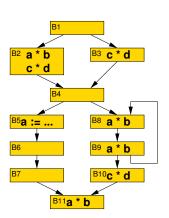
. . . .

E-Pfad PRE

Notation

E-Pfade





Für a\*b

- $[b_8, b_9]$
- $[b_9, b_8]$
- $\bullet$  [ $b_9$ ,  $b_{10}$ ,  $b_{11}$ ]
- Nicht:  $[b_2, b_4, b_8]$
- Für c\*d
  - $[b_2, b_4, b_8, b_9, b_{10}]$ •  $[b_3, b_4, b_8, b_9, b_{10}]$

OC

A. Kocn

inleitung

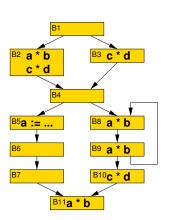
...

F-Pfad PRF

Notation

E-Pfade





Für a\*b

- $[b_8, b_9]$
- $[b_9, b_8]$
- $\bullet$  [ $b_9$ ,  $b_{10}$ ,  $b_{11}$ ]
- Nicht:  $[b_2, b_4, b_8]$
- Für c\*d
  - $\bullet$  [ $b_2$ ,  $b_4$ ,  $b_8$ ,  $b_9$ ,  $b_{10}$ ]
  - $[b_3, b_4, b_8, b_9, b_{10}]$

ОС

A. Koch

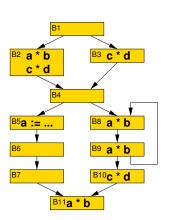
inleitung

. . . \_

E-Pfad PRE

Notation





Für a\*b

•  $[b_8, b_9]$ 

•  $[b_9, b_8]$ 

•  $[b_9, b_{10}, b_{11}]$ 

• Nicht:  $[b_2, b_4, b_8]$ 

• Für c\*d

•  $[b_2, b_4, b_8, b_9, b_{10}]$ •  $[b_3, b_4, b_8, b_9, b_{10}]$  OC

A. Kocn

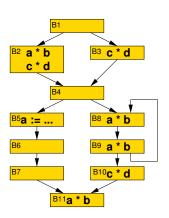
Einleitung

/annanta

E-Pfad PRE

Notation





Für a\*b

•  $[b_8, b_9]$ 

•  $[b_9, b_8]$ 

•  $[b_9, b_{10}, b_{11}]$ 

• Nicht:  $[b_2, b_4, b_8]$ 

• Für c\*d

[b<sub>2</sub>, b<sub>4</sub>, b<sub>8</sub>, b<sub>9</sub>, b<sub>10</sub>]
[b<sub>3</sub>, b<sub>4</sub>, b<sub>8</sub>, b<sub>9</sub>, b<sub>10</sub>]

OC

A. Koch

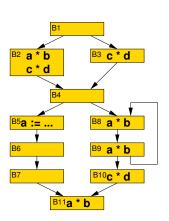
Einleitung

.

E-Pfad PRE

Notation





Für a\*b

•  $[b_8, b_9]$ 

•  $[b_9, b_8]$ 

•  $[b_9, b_{10}, b_{11}]$ 

• Nicht:  $[b_2, b_4, b_8]$ 

Für c\*d

 $\bullet$   $[b_2, b_4, b_8, b_9, b_{10}]$ 

 $\bullet$  [ $b_3$ ,  $b_4$ ,  $b_8$ ,  $b_9$ ,  $b_{10}$ ]

OC

A. Koch

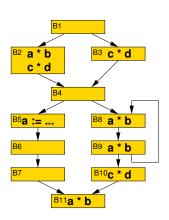
inleitung

Conzente

E-Pfad PRE

Notation





Für a\*b

•  $[b_8, b_9]$ 

•  $[b_9, b_8]$ 

•  $[b_9, b_{10}, b_{11}]$ 

• Nicht:  $[b_2, b_4, b_8]$ 

• Für c\*d

•  $[b_2, b_4, b_8, b_9, b_{10}]$ 

 $\bullet$  [ $b_3$ ,  $b_4$ ,  $b_8$ ,  $b_9$ ,  $b_{10}$ ]

OC

A. Koch

Einleitung

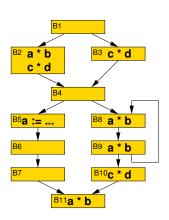
/----

E-Pfad PRE

Notation

E-Plade





• Für a\*b

•  $[b_8, b_9]$ 

•  $[b_9, b_8]$ 

•  $[b_9, b_{10}, b_{11}]$ 

• Nicht:  $[b_2, b_4, b_8]$ 

• Für c\*d

•  $[b_2, b_4, b_8, b_9, b_{10}]$ 

•  $[b_3, b_4, b_8, b_9, b_{10}]$ 

OC

A. Koch

inleitung

Conzente

E-Pfad PRE

FR



- Das erste Vorkommen von e im Endblock  $b_k$  des E-Pfades  $[b_i, \ldots, b_k]$  ist eliminierbar
- Das vorherige Evaluationsergebnis des Ausdrucks e wird dazu gesichert in der Variable  $t_e$
- Wird der Pfad  $(b_i, \ldots, b_k)$  von einem Block ausserhalb  $b_h$  über eine Kante  $(b_h, b_j)$  betreten
- ... muss e (falls nötig) mit  $t_e := e$  berechnet und gesichert werden ...
  - Am Ende des Blocks  $b_n$ , falls  $|\operatorname{succ}(b_n)| = 1$ 
    - Auf der aufgeteilten Kante  $(b_h, b_j)$  sonst
  - Lebenszeitoptimale Platzierung (so spät wie möglich)
- Eine Neuberechnung und Sicherung ist nicht nötig, falls  $b_h$  selber auf einem E-Pfad für e liegt.



A. Koch

inleitung

onzepte

E-Pfad PRE

Notation

-Ptade



- Das erste Vorkommen von e im Endblock  $b_k$  des E-Pfades  $[b_i, \ldots, b_k]$  ist eliminierbar
- Das vorherige Evaluationsergebnis des Ausdrucks e wird dazu gesichert in der Variable  $t_e$
- Wird der Pfad  $(b_i, \ldots, b_k)$  von einem Block ausserhalb  $b_h$  über eine Kante  $(b_h, b_i)$  betreten
- ... muss e (falls nötig) mit  $t_e := e$  berechnet und gesichert werden ...
  - Am Ende des Blocks b<sub>li</sub>, falls |succ(b<sub>h</sub>)| = 1
    Auf der aufgeteilten Kante (b<sub>h</sub>, b<sub>j</sub>) sonst
    Lebenszeitoptimale Platzierung (so spät wie möglich)
- Eine Neuberechnung und Sicherung ist nicht nötig, falls  $b_h$  selber auf einem E-Pfad für e liegt.

UU

A. Koch

inleitung

KE

onzepte

E-Pfad PRE

-Pfade

atennuss



- Das erste Vorkommen von e im Endblock  $b_k$  des E-Pfades  $[b_i, \ldots, b_k]$  ist eliminierbar
- Das vorherige Evaluationsergebnis des Ausdrucks e wird dazu gesichert in der Variable t<sub>e</sub>
- Wird der Pfad  $(b_i, ..., b_k)$  von einem Block ausserhalb  $b_h$  über eine Kante  $(b_h, b_j)$  betreten
- ... muss e (falls nötig) mit  $t_e := e$  berechnet und gesichert werden ...
  - Am Ende des Blocks b<sub>h</sub>, falls |succ(b<sub>h</sub>)| = 1
    Auf der aufgeteilten Kante (b<sub>h</sub>, b<sub>j</sub>) sonst
    Lebenszeitoptimale Platzierung (so spät wie möglich)
- Eine Neuberechnung und Sicherung ist nicht nötig, falls  $b_h$  selber auf einem E-Pfad für e liegt.



A. Koch

inieitung

'KE

Conzepte

E-Pfad PRE

otation

E-Ptade

Jaleilluss



- Das erste Vorkommen von e im Endblock  $b_k$  des E-Pfades  $[b_i, \ldots, b_k]$  ist eliminierbar
- Das vorherige Evaluationsergebnis des Ausdrucks e wird dazu gesichert in der Variable  $t_e$
- Wird der Pfad  $(b_i, ..., b_k)$  von einem Block ausserhalb  $b_h$  über eine Kante  $(b_h, b_j)$  betreten
- ... muss e (falls nötig) mit t<sub>e</sub> := e berechnet und gesichert werden ...
  - Am Ende des Blocks  $b_h$ , falls  $|\operatorname{succ}(b_h)| = 1$
  - Auf der aufgeteilten Kante  $(b_h, b_j)$  sonst
  - Lebenszeitoptimale Platzierung (so spät wie möglich)
- Eine Neuberechnung und Sicherung ist nicht nötig, falls  $b_h$  selber auf einem E-Pfad für e liegt.



A. Koch

inleitung

'KE

Conzepte

E-Pfad PRE

-Pfade

\_-i laue



- Das erste Vorkommen von e im Endblock  $b_k$  des E-Pfades  $[b_i, \ldots, b_k]$  ist eliminierbar
- Das vorherige Evaluationsergebnis des Ausdrucks e wird dazu gesichert in der Variable  $t_e$
- Wird der Pfad  $(b_i, \ldots, b_k)$  von einem Block ausserhalb  $b_h$  über eine Kante  $(b_h, b_j)$  betreten
- ... muss e (falls nötig) mit t<sub>e</sub> := e berechnet und gesichert werden ...
  - Am Ende des Blocks  $b_h$ , falls  $|\operatorname{succ}(b_h)| = 1$
  - Auf der aufgeteilten Kante  $(b_h, b_j)$  sonst
  - Lebenszeitoptimale Platzierung (so spät wie möglich)
- Eine Neuberechnung und Sicherung ist nicht nötig, falls  $b_h$  selber auf einem E-Pfad für e liegt.



A. Koch

inieitung

'KE

Conzepte

E-Pfad PRE

-Pfade





- Das erste Vorkommen von e im Endblock  $b_k$  des E-Pfades  $[b_i, \ldots, b_k]$  ist eliminierbar
- Das vorherige Evaluationsergebnis des Ausdrucks e wird dazu gesichert in der Variable  $t_e$
- Wird der Pfad  $(b_i, ..., b_k)$  von einem Block ausserhalb  $b_h$  über eine Kante  $(b_h, b_j)$  betreten
- ... muss e (falls nötig) mit t<sub>e</sub> := e berechnet und gesichert werden ...
  - Am Ende des Blocks  $b_h$ , falls  $|\operatorname{succ}(b_h)| = 1$
  - Auf der aufgeteilten Kante  $(b_h, b_j)$  sonst
  - Lebenszeitoptimale Platzierung (so spät wie möglich)
- Eine Neuberechnung und Sicherung ist nicht nötig, falls  $b_h$  selber auf einem E-Pfad für e liegt.



A. Koch

inleitung

'KE

conzepte

E-Pfad PRE

otation

E-Pfade

Dateriius



- Das erste Vorkommen von e im Endblock  $b_k$  des E-Pfades  $[b_i, \ldots, b_k]$  ist eliminierbar
- Das vorherige Evaluationsergebnis des Ausdrucks e wird dazu gesichert in der Variable  $t_e$
- Wird der Pfad  $(b_i, ..., b_k)$  von einem Block ausserhalb  $b_h$  über eine Kante  $(b_h, b_j)$  betreten
- ... muss e (falls nötig) mit t<sub>e</sub> := e berechnet und gesichert werden ...
  - Am Ende des Blocks  $b_h$ , falls  $|\operatorname{succ}(b_h)| = 1$
  - Auf der aufgeteilten Kante  $(b_h, b_j)$  sonst
  - Lebenszeitoptimale Platzierung (so spät wie möglich)
- Eine Neuberechnung und Sicherung ist nicht nötig, falls  $b_h$  selber auf einem E-Pfad für e liegt.



A. Koch

inentung

I L

conzepte

E-Pfad PRE

otation



- Das erste Vorkommen von e im Endblock  $b_k$  des E-Pfades  $[b_i, \ldots, b_k]$  ist eliminierbar
- Das vorherige Evaluationsergebnis des Ausdrucks e wird dazu gesichert in der Variable  $t_e$
- Wird der Pfad  $(b_i, \ldots, b_k)$  von einem Block ausserhalb  $b_h$  über eine Kante  $(b_h, b_j)$  betreten
- ... muss e (falls nötig) mit t<sub>e</sub> := e berechnet und gesichert werden ...
  - Am Ende des Blocks  $b_h$ , falls  $|\operatorname{succ}(b_h)| = 1$
  - Auf der aufgeteilten Kante  $(b_h, b_j)$  sonst
  - Lebenszeitoptimale Platzierung (so spät wie möglich)
- Eine Neuberechnung und Sicherung ist nicht nötig, falls  $b_h$  selber auf einem E-Pfad für e liegt.



A. Koch

inleitung

---

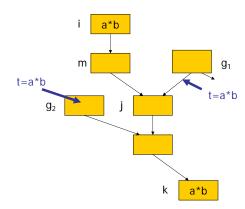
F-Pfad PRF

otation

E-Pfade

# Lebenszeitoptimalität durch E-Pfade





ОС

A. Koch

inleitung

Konzent

E-Pfad PRE

Notation

E-Pfade

atenfluss

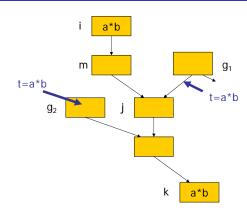
• E-Pfad: [*i*,...,*k*]

• Einfügen von Berechnungen von e und Kopieren

◆ロト ◆個ト ◆量ト ◆量ト ■ 少久(\*)

# Lebenszeitoptimalität durch E-Pfade





OC

A. NOCH

inleitung

FNE

Konzepte

E-Pfad PRE

E-Pfade

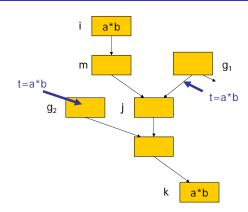
atenfluss

- E-Pfad: [i,...,k]
- Einfügen von Berechnungen von e und Kopieren

• In Kante  $(g_1, j)$  und Block  $g_2$ 

# Lebenszeitoptimalität durch E-Pfade





ОС

A. NOCH

inleitung

Konzente

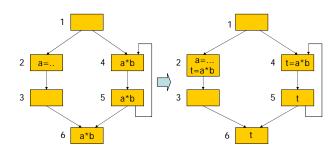
F-Pfad PRF

Notation

E-Pfade

- E-Pfad:  $[i, \ldots, k]$
- Einfügen von Berechnungen von e und Kopieren
  - In Kante  $(g_1, j)$  und Block  $g_2$





• E-Pfade: [4,5], [5,4], [5,6]

Herstellen totaler Redundanz: Einfügen in Ast 2,3
 Nach MRA in 2, mit E-Pfaden in 3: Lebenszeitoptimal

Entfernen von Redundanz in 4

Nach MRA nicht möglich

[5,4] ist E-Ptad, Berechnung in 4 muss entternbar sein
 → Kante (1,4) aufteilen, dort Berechnen und Kopieren

OC

A. NOCI

leitung

RE

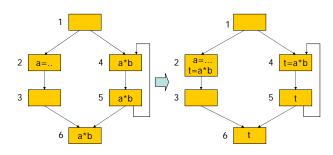
Konzepte

E-Pfad PRE

\_ \_ . . .

E-Plade





• E-Pfade: [4,5], [5,4], [5,6]

Herstellen totaler Redundanz: Einfügen in Ast 2,3
 Nach MRA in 2, mit E-Pfaden in 3: Lebenszeitoptimal

Entfernen von Redundanz in 4

Nach MRA nicht moglich
 [5,4] ist E-Pfad, Berechnung

[5,4] Ist E-Plad, Berechnung in 4 muss entiernoar sein
 → Kante (1,4) aufteilen, dort Berechnen und Kopieren



A. Kocn

nleitung

PRE

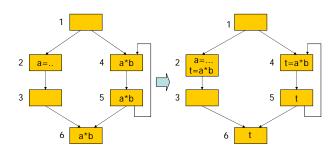
F-Pfad PRF

Notation

E-Pfade



F-Pfad PRF



• E-Pfade: [4,5], [5,4], [5,6]

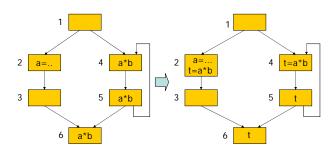
- Herstellen totaler Redundanz: Einfügen in Ast 2,3
  - Nach MRA in 2, mit E-Pfaden in 3: Lebenszeitoptimal
- Entfernen von Redundanz in 4

Nach MRA nicht möglich

[5,4] ist E-Pfad, Berechnung in 4 muss entfernbar sein
 → Kante (1.4) aufteilen, dort Berechnen und Kopieren







- E-Pfade: [4,5], [5,4], [5,6]
- Herstellen totaler Redundanz: Einfügen in Ast 2,3
  - Nach MRA in 2, mit E-Pfaden in 3: Lebenszeitoptimal
- Entfernen von Redundanz in 4
  - Nach MRA nicht möglich
  - [5,4] ist E-Pfad, Berechnung in 4 muss entfernbar sein
     → Kante (1,4) aufteilen, dort Berechnen und Kopieren



A. Koch

nleitung

Konzepte

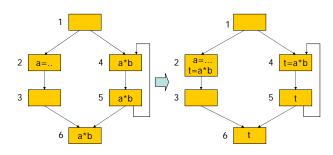
E-Pfad PRE

- -

Detenflue

Dateriius





• E-Pfade: [4,5], [5,4], [5,6]

- Herstellen totaler Redundanz: Einfügen in Ast 2,3
  - Nach MRA in 2, mit E-Pfaden in 3: Lebenszeitoptimal
- Entfernen von Redundanz in 4
  - Nach MRA nicht möglich
  - [5,4] ist E-Pfad, Berechnung in 4 muss entfernbar sein  $\rightarrow$  Kante (1,4) aufteilen, dort Berechnen und Kopieren



A. Koch

inleitung RE

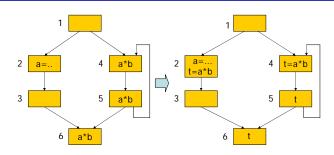
Konzepte

E-Pfad PRE

- -

E-Plaue





- E-Pfade: [4,5], [5,4], [5,6]
- Herstellen totaler Redundanz: Einfügen in Ast 2,3
  - Nach MRA in 2, mit E-Pfaden in 3: Lebenszeitoptimal
- Entfernen von Redundanz in 4
  - Nach MRA nicht möglich
  - [5,4] ist E-Pfad, Berechnung in 4 muss entfernbar sein
    - ightarrow Kante (1,4) aufteilen, dort Berechnen und Kopieren



A. Koch

inleitung RE

Konzepte

E-Pfad PRE

F-Pfade

z-Maue



Wert von e sichern

Füge Anweisung  $\mathbf{t}_e := e$  vor einer Benutzung von e ein und ersetze Benutzung durch  $t_e$ 

Neue Berechnung von e einfügen Füge Anweisung  $t_e := e$  ein

edundante Berechnung von e

Ersetze e durch  $t_a$ 

An welchen Stellen diese Aktionen ausführen?



A. Koch

nreitung

Konzepte

E-Pfad PRE

inotation

E-Prade





Wert von e sichern

Füge Anweisung  $\mathbf{t}_e := e$  vor einer Benutzung von e ein und ersetze Benutzung durch  $t_e$ 

Neue Berechnung von e einfügen

Füge Anweisung  $t_e := e ein$ 

Eliminiere redundante Berechnung von eErsetze e durch  $t_e$ 

An welchen Stellen diese Aktionen ausführen?

ОС

A. Koch

DE

Konzepte F-Pfad PRF

= D( )



Wert von e sichern

Füge Anweisung  $t_e := e$  vor einer Benutzung von e ein und ersetze Benutzung durch  $t_e$ 

Neue Berechnung von e einfügen

Füge Anweisung  $t_e := e ein$ 

Eliminiere redundante Berechnung von e

Ersetze e durch  $t_e$ 

An welchen Stellen diese Aktionen ausführen?



A. Koch

ınleitung

Konzepte

E-Pfad PRE

inotation

E-Piade



Wert von e sichern

Füge Anweisung  $t_e := e$  vor einer Benutzung von e ein und ersetze Benutzung durch  $t_e$ 

Neue Berechnung von e einfügen

Füge Anweisung  $t_e := e ein$ 

Eliminiere redundante Berechnung von e

Ersetze e durch  $t_e$ 

An welchen Stellen diese Aktionen ausführen?



A. Koch

ınleitung

Konzepte

E-Pfad PRE

inotation

E-Piade

## Aktionen während Optimierung



Wert von e sichern

Füge Anweisung  $\mathbf{t}_e := e$  vor einer Benutzung von e ein und ersetze Benutzung durch  $t_e$ 

Neue Berechnung von e einfügen

Füge Anweisung  $t_e := e ein$ 

Eliminiere redundante Berechnung von e

Ersetze e durch  $t_e$ 

An welchen Stellen diese Aktionen ausführen?



A. Koch

inleitung

Konzepte

E-Pfad PRE

Notation

E-Piade



## **Neue Notation**

OC

I. Koch

RE

onzepte.

Notation

-Pfade

Alternative zur Mengennotation



Prädikate bestimmen für jeden Ausdruck e, ob eine Aussage wahr oder falsch ist

Wahr für alle am Ende eines Blocks b verfügbaren Ausdr. avail(b) = TRUE für  $e \Leftrightarrow e \in \mathsf{AVAILOUT}(b)$ 

Wahr für alle in den Block b vorziehbaren Ausdrücke ant(b) = TRUE für  $e \Leftrightarrow e \in \mathsf{ANT}(b)$ 

Wahr für alle Ausdrücke, für die beides gilt avail $(b) \cdot \operatorname{ant}(b) = \operatorname{TRUE} \Leftrightarrow e \in (\operatorname{AVAILOUT}(b) \cap \operatorname{ANT}(b))$ 

Wahr für alle Ausdrücke, für die mindestens eines gilt avail(b) + ant(b) = TRUE  $\Leftrightarrow e \in (\mathsf{AVAILOUT}(b) \cup \mathsf{ANT}(b))$ 

OC

A. Koch

ileitung IE

-Pfad PRE

Notation

-Pfade

Alternative zur Mengennotation

embedded systems + applications

Prädikate bestimmen für jeden Ausdruck e, ob eine Aussage wahr oder falsch ist

Wahr für alle am Ende eines Blocks  $\emph{b}$  verfügbaren Ausdr.

 $\mathsf{avail}(\mathit{b}) = \mathsf{TRUE}\ \mathsf{f\"{u}r}\ e \Leftrightarrow e \in \mathsf{AVAILOUT}(b)$ 

Wahr für alle in den Block b vorziehbaren Ausdrücke ant(b) = TRUE für  $e \Leftrightarrow e \in \mathsf{ANT}(b)$ 

Wahr für alle Ausdrücke, für die beides gilt avail $(b) \cdot \operatorname{ant}(b) = \operatorname{TRUE} \Leftrightarrow e \in (\operatorname{AVAILOUT}(b) \cap \operatorname{ANT}(b))$ 

Wahr für alle Ausdrücke, für die mindestens eines gilt avail(b) + ant(b) = TRUE  $\Leftrightarrow e \in (AVAILOUT(b) \cup ANT(b)$ 

OC

A. Koch

nleitung RE

onzepte

Notation

-Pfade

Alternative zur Mengennotation

embedded systems + applications

Prädikate bestimmen für jeden Ausdruck e, ob eine Aussage wahr oder falsch ist

Wahr für alle am Ende eines Blocks b verfügbaren Ausdr.

 $\mathsf{avail}(\mathit{b}) = \mathsf{TRUE}\ \mathsf{f\"{u}r}\ e \Leftrightarrow e \in \mathsf{AVAILOUT}(b)$ 

Wahr für alle in den Block b vorziehbaren Ausdrücke ant(b) = TRUE für  $e \Leftrightarrow e \in ANT(b)$ 

Wahr für alle Ausdrücke, für die beides gilt  $\mathsf{avail}(b) \cdot \mathsf{ant}(b) = \mathsf{TRUE} \Leftrightarrow e \in (\mathsf{AVAILOUT}(b) \cap \mathsf{ANT}(b))$ 

Wahr für alle Ausdrücke, für die mindestens eines gilt avail(b) + ant(b) = TRUE  $\Leftrightarrow e \in (AVAILOUT(b) \cup ANT(b)$ 

OC

A. Koch

nleitung RE

onzepte

Notation

-Pfade

Alternative zur Mengennotation

embedded systems + applications

Prädikate bestimmen für jeden Ausdruck e, ob eine Aussage wahr oder falsch ist

Wahr für alle am Ende eines Blocks b verfügbaren Ausdr.

 $avail(b) = TRUE für e \Leftrightarrow e \in AVAILOUT(b)$ 

Wahr für alle in den Block *b* vorziehbaren Ausdrücke

 $ant(b) = TRUE für e \Leftrightarrow e \in Ant(b)$ 

Wahr für alle Ausdrücke, für die beides gilt

 $avail(b) \cdot ant(b) = TRUE \Leftrightarrow e \in (AVAILOUT(b) \cap ANT(b))$ 

Wahr für alle Ausdrücke, für die mindestens eines gilt avail(b) + ant(b) = TRUE  $\Leftrightarrow e \in (AVAILOUT(b) \cup ANT(b))$ 

OC

A. Koch

nleitung RE

onzepte

Notation

E-Pfade

#### Prädikatenschreibweise Alternative zur Mengennotation

Prädikate bestimmen für jeden Ausdruck *e*, ob eine Aussage wahr oder falsch ist

Wahr für alle am Ende eines Blocks b verfügbaren Ausdr.

avail(b) = TRUE für  $e \Leftrightarrow e \in AVAILOUT(b)$ 

Wahr für alle in den Block b vorziehbaren Ausdrücke

 $ant(b) = TRUE für e \Leftrightarrow e \in Ant(b)$ 

Wahr für alle Ausdrücke, für die beides gilt

 $avail(b) \cdot ant(b) = TRUE \Leftrightarrow e \in (AvailOut(b) \cap Ant(b))$ 

Wahr für alle Ausdrücke, für die mindestens eines gilt

 $avail(b) + ant(b) = TRUE \Leftrightarrow e \in (AVAILOUT(b) \cup ANT(b))$ 

A Koch

Notation

## Konfluenzoperatoren in Prädikatenschreibweise



#### Konjunktion

$$\begin{array}{l} \textit{cpin(b)} = \prod_{p} \textit{cpout(p)} = \mathsf{TRUE} \; \mathsf{für} \; \mathsf{Kopie} \; c \\ \Leftrightarrow c \in \bigcap_{p \in \textit{pred(b)}} \mathsf{CPout}(p) \end{array}$$

#### Disjunktion

```
liveout(b) = \sum_{s} livein(s) = TRUE  für Variable v \Leftrightarrow v \in \bigcup_{s \in Succ(b)} LiveIn(s)
```

OC

1. 110011

ileiturig RE

onzepte

Notation

E-Pfade

## Konfluenzoperatoren in Prädikatenschreibweise



#### Konjunktion

$$\begin{array}{l} \textit{cpin(b)} = \prod_{p} \textit{cpout(p)} = \mathsf{TRUE} \; \mathsf{für} \; \mathsf{Kopie} \; c \\ \Leftrightarrow c \in \bigcap_{p \in \textit{pred(b)}} \mathsf{CPout}(p) \end{array}$$

#### Disjunktion

 $\begin{array}{l} \textit{liveout(b)} = \sum_{s} \textit{livein(s)} = \mathsf{TRUE} \; \mathsf{für} \; \mathsf{Variable} \; v \\ \Leftrightarrow v \in \bigcup_{s \in \textit{Succ(b)}} \mathsf{LiveIn}(s) \end{array}$ 

OC

A. NOCH

nieitung RE

onzepte

Notation

E-Pfade



## Für alle Ausdrücke eindeutige Zuordnung festlegen

$$\mathbf{a}^*\mathbf{b} \to \mathrm{Bit} \ \mathbf{1}$$
 $\mathbf{c}^*\mathbf{d} \to \mathrm{Bit} \ \mathbf{2}$ 

Wert	entspricht	entspricht
12	Prädikat x	Menge
00	$\neg x_{a*b}, \neg x_{c*d}$	Ø
01	$\neg x_{a*b}, x_{c*d}$	$\{c*d\}$
10	$x_{a*b}, \neg x_{c*d}$	$\{a*b\}$
11	$x_{a*b}, x_{c*d}$	$\{a*b,c*d\}$

- Bitvektoren können sehr lang werden
- Prädikate liefern Ergebnisse als Bitvektor
- Damit logische Verknüpfungen effizient

ОС

A. Koch

RE

onzepte

Notation

-Pfade

-i lauc



## Für alle Ausdrücke eindeutige Zuordnung festlegen

$$\mathbf{a}^*\mathbf{b} o \mathsf{Bit} \ 1$$
  
 $\mathbf{c}^*\mathbf{d} o \mathsf{Bit} \ 2$ 

• •

Wert	entspricht	entspricht
12	Prädikat x	Menge
00	$\neg x_{a*b}, \neg x_{c*d}$	Ø
01	$\neg x_{a*b}, x_{c*d}$	$\{\mathbf{c} * \mathbf{d}\}$
10	$x_{a*b}, \neg x_{c*d}$	$\{a*b\}$
11	$x_{a*b}, x_{c*d}$	$\{a*b,c*d\}$

- Bitvektoren können sehr lang werden
- Prädikate liefern Ergebnisse als Bitvekto

ОС

A. Kocn

RF

onzepte

Pfad PRI

Notation

-Pfade



#### Für alle Ausdrücke eindeutige Zuordnung festlegen

$$\mathbf{a}^*\mathbf{b} \to \mathrm{Bit} \ \mathbf{1}$$
 $\mathbf{c}^*\mathbf{d} \to \mathrm{Bit} \ \mathbf{2}$ 

Wert	entspricht	entspricht
12	Prädikat x	Menge
00	$\neg x_{a*b}, \neg x_{c*d}$	Ø
01	$\neg x_{a*b}, x_{c*d}$	$\{\mathbf{c} * \mathbf{d}\}$
10	$x_{a*b}, \neg x_{c*d}$	$\{\mathbf{a} * \mathbf{b}\}$
11	$x_{a*b}, x_{c*d}$	$\{a*b,c*d\}$

- Bitvektoren können sehr lang werden
- Pradikate liefern Ergebnisse als Bitvektor
- Damit logische Verknüpfungen effizient

implementierbar 4 P + 4



A. Koch

Inleitung

Konzepte

E-Pfad PR

Notation

-Pfade

vatentiuss



#### Für alle Ausdrücke eindeutige Zuordnung festlegen

$$\mathbf{a}^*\mathbf{b} \to \mathsf{Bit} \ \mathbf{1}$$
 $\mathbf{c}^*\mathbf{d} \to \mathsf{Bit} \ \mathbf{2}$ 

Wert	entspricht	entspricht
12	Prädikat x	Menge
00	$\neg x_{a*b}, \neg x_{c*d}$	Ø
01	$\neg x_{a*b}, x_{c*d}$	$\{\mathbf{c} * \mathbf{d}\}$
10	$x_{a*b}, \neg x_{c*d}$	$\{\mathbf{a} * \mathbf{b}\}$
11	$\chi_{a*h}, \chi_{c*d}$	$\{a*b,c*d\}$

- Bitvektoren können sehr lang werden
- Prädikate liefern Ergebnisse als Bitvektor
- Damit logische Verknüpfungen effizient

  implementischen



A. Koch

inleitung

onzepte

Notation

-Pfade



#### Für alle Ausdrücke eindeutige Zuordnung festlegen

$$\mathbf{a^*b} \to \mathsf{Bit} \ 1$$
  
 $\mathbf{c^*d} \to \mathsf{Bit} \ 2$ 

Wertentsprichtentspricht12Prädikat xMenge00 $\neg x_{a*b}$ ,  $\neg x_{c*d}$  $\emptyset$ 01 $\neg x_{a*b}$ ,  $x_{c*d}$  $\{\mathbf{c} * \mathbf{d}\}$ 10 $x_{a*b}$ ,  $\neg x_{c*d}$  $\{\mathbf{a} * \mathbf{b}\}$ 11 $x_{a*h}$ ,  $x_{c*d}$  $\{\mathbf{a} * \mathbf{b}, \mathbf{c} * \mathbf{d}\}$ 

- Bitvektoren können sehr lang werden
- Prädikate liefern Ergebnisse als Bitvektor
- Damit logische Verknüpfungen effizient implementierhar



A. Koch

inleitung

Conzepte

Notation

-Pfade

-Piade



#### Für alle Ausdrücke eindeutige Zuordnung festlegen

$$\mathbf{a}^{\mathbf{*}}\mathbf{b} \to \mathsf{Bit} \ \mathbf{1}$$
  $\mathbf{c}^{\mathbf{*}}\mathbf{d} \to \mathsf{Bit} \ \mathbf{2}$ 

Wertentsprichtentspricht12Prädikat xMenge00 $\neg x_{a*b}$ ,  $\neg x_{c*d}$  $\emptyset$ 01 $\neg x_{a*b}$ ,  $x_{c*d}$  $\{\mathbf{c} * \mathbf{d}\}$ 10 $x_{a*b}$ ,  $\neg x_{c*d}$  $\{\mathbf{a} * \mathbf{b}\}$ 11 $x_{a*b}$ ,  $x_{c*d}$  $\{\mathbf{a} * \mathbf{b}, \mathbf{c} * \mathbf{d}\}$ 

- Bitvektoren können sehr lang werden
- Prädikate liefern Ergebnisse als Bitvektor
- Damit logische Verknüpfungen effizient implementierbar



A. Koch

RE

onzepte

Notation

-Pfade

# Beispiel: Verküpfung von Prädikaten Mittels Bit-Operatoren



#### Zuordnung von Ausdrücken e an Bits

$$\mathbf{a^*b} o \mathrm{Bit} \ 1$$
 $\mathbf{c^*d} o \mathrm{Bit} \ 2$ 
 $\mathbf{a^*d} o \mathrm{Bit} \ 3$ 

Gegeben: Zwei Prädikate x und y bezüglich Ausdruck eAnnahme: x=101, y=011

$$x \cdot y = x & y = 001$$
  
 $x + y = x | y = 111$ 

OC

I. Koch

nieituri

onzepte

Notation

-Pfade

# Beispiel: Verküpfung von Prädikaten Mittels Bit-Operatoren



Zuordnung von Ausdrücken e an Bits

$$\mathbf{a^*b} o \operatorname{Bit} 1$$
 $\mathbf{c^*d} o \operatorname{Bit} 2$ 
 $\mathbf{a^*d} o \operatorname{Bit} 3$ 

. . .

Gegeben: Zwei Prädikate x und y bezüglich Ausdruck e Annahme: x=101,y=011

$$x \cdot y = x & y = 001$$
$$x + y = x | y = 111$$



A. Koch

nleitung

onzepte

Notation

-Pfade

# Beispiel: Verküpfung von Prädikaten Mittels Bit-Operatoren



Zuordnung von Ausdrücken e an Bits

$$\mathbf{a^*b} o \operatorname{Bit} 1$$
 $\mathbf{c^*d} o \operatorname{Bit} 2$ 
 $\mathbf{a^*d} o \operatorname{Bit} 3$ 

Gegeben: Zwei Prädikate x und y bezüglich Ausdruck e

Annahme: x = 101, y = 011

$$x \cdot y = x & y = 001$$
$$x + y = x | y = 111$$



A. Koch

inleitung RF

onzepte

Notation

-Pfade





# Bestimmen von E-Pfaden

OC

A. Koch

RE

Konzepte

E-Piad PRI

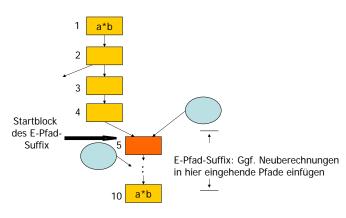
Notation

E-Pfade

#### Anatomie eines E-Pfades



F-Pfade



 $b \in (1,2)$ 

 $avail(b) \cdot \neg ant(b)$ 

 $b \in (3,4)$ 

avail(b) ant(b)

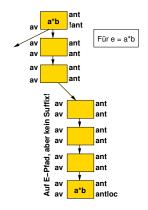
 $\neg$ avail(b) · ant(b)

E-Pfad-Suffix

# Beispiel: Leerer E-Pfad-Suffix



Anforderung an E-Pfad-Suffix:  $\neg$ avail(b)  $\cdot$  ant(b)



OC

A. Koch

inleitung \_ \_

Konzepte

E-Pfad PR

NOLALION

E-Pfade



- E-Pfad  $[b_i, \ldots, b_k]$  hat drei (ggf. leere) Segmente
  - avail(b) ·¬ant(b)
  - avail(b) · ant(b)
  - ¬avail(b) · ant(b)
- Finde Start des E-Pfad-Suffixes: Suche Block  $b_m$  mit  $\neg$ avail $(b_m) \cdot$  ant $(b_m) \cdot \sum_v$  avail(p)
- Suche rückwärts von b<sub>m</sub> durch Segmente avail(b) · ant(b)
   avail(b) · ¬ant(b)
   bis Start des E-Pfads b<sub>i</sub> erreicht
- Suche vorwärts von b<sub>m</sub> durch Segment
   ¬avail(b) · ant(b)
   bis Ende des E-Pfades erreicht



A. Kocn

inleitung

Konzenta

Dfod DDE

Notation

E-Pfade



- E-Pfad  $[b_i, \ldots, b_k]$  hat drei (ggf. leere) Segmente
  - avail(b) ·¬ant(b)
  - avail(b) · ant(b)
  - ¬avail(b) · ant(b)
- Finde Start des E-Pfad-Suffixes: Suche Block  $b_m$  mit  $\neg$ avail $(b_m) \cdot$  ant $(b_m) \cdot \sum_v$  avail(p)
- Suche rückwärts von b<sub>m</sub> durch Segmente avail(b) · ant(b) avail(b) · ¬ant(b) bis Start des E-Pfads b<sub>i</sub> erreicht
- Suche vorwärts von b<sub>m</sub> durch Segment
   ¬avail(b) · ant(b)
   bis Ende des E-Pfades erreicht



A. KOCH

inleitung

Konzente

F-Pfad PRI

Notation

E-Pfade



- E-Pfad  $[b_i, \ldots, b_k]$  hat drei (ggf. leere) Segmente
  - avail(b) ·¬ant(b)
  - avail(b) · ant(b)
  - ¬avail(b) · ant(b)
- Finde Start des E-Pfad-Suffixes: Suche Block  $b_m$  mit  $\neg$ avail $(b_m) \cdot$  ant $(b_m) \cdot \sum_v$  avail(p)
- Suche rückwärts von b<sub>m</sub> durch Segmente avail(b) · ant(b) avail(b) · ¬ant(b) bis Start des E-Pfads b<sub>i</sub> erreicht
- Suche vorwärts von b<sub>m</sub> durch Segment
   ¬avail(b) · ant(b)
   bis Ende des E-Pfades erreicht



A. Koch

inleitung

PRE

Konzepte

-Plau PRE

Notation

E-Pfade



- E-Pfad  $[b_i, \ldots, b_k]$  hat drei (ggf. leere) Segmente
  - avail(b) ·¬ant(b)
  - avail(b) · ant(b)
  - ¬avail(b) · ant(b)
- Finde Start des E-Pfad-Suffixes: Suche Block  $b_m$  mit  $\neg$ avail $(b_m) \cdot \text{ant}(b_m) \cdot \sum_v \text{avail}(p)$
- Suche rückwärts von b<sub>m</sub> durch Segmente avail(b) · ant(b) avail(b) · ¬ant(b) bis Start des E-Pfads b<sub>i</sub> erreicht
- Suche vorwärts von b<sub>m</sub> durch Segment
   ¬avail(b) · ant(b)
   bis Ende des E-Pfades erreicht



A. KOCII

Einleitung

'RE

vousebre

- 1 100 1 112

F-Pfade

z-Plade



- E-Pfad  $[b_i, \ldots, b_k]$  hat drei (ggf. leere) Segmente
  - avail(b) ·¬ant(b)
  - avail(b) · ant(b)
  - ¬avail(b) · ant(b)
- Finde Start des E-Pfad-Suffixes: Suche Block  $b_m$  mit  $\neg$ avail $(b_m) \cdot$  ant $(b_m) \cdot \sum_v$  avail(p)
- Suche rückwärts von b<sub>m</sub> durch Segmente avail(b) · ant(b) avail(b) · ¬ant(b) bis Start des E-Pfads b<sub>i</sub> erreicht



A. KOCH

Einleitung

RE

Konzepte

\_ . ....

F-Pfade

E-Ptade



- E-Pfad  $[b_i, \ldots, b_k]$  hat drei (ggf. leere) Segmente
  - avail(b) ·¬ant(b)
  - avail(b) · ant(b)
  - ¬avail(b) · ant(b)
- Finde Start des E-Pfad-Suffixes: Suche Block  $b_m$  mit  $\neg$ avail $(b_m) \cdot$  ant $(b_m) \cdot \sum_p$  avail(p)
- Suche rückwärts von b<sub>m</sub> durch Segmente avail(b) · ant(b) avail(b) · ¬ant(b) bis Start des E-Pfads b<sub>i</sub> erreicht
- Suche vorwärts von b<sub>m</sub> durch Segment
   ¬avail(b) · ant(b)
   bis Ende des E-Pfades erreicht



A. NOCH

Einleitung

'KE

Konzepte

E-FIAU FNI

Notation

E-Pfade



- E-Pfad  $[b_i, \ldots, b_k]$  hat drei (ggf. leere) Segmente
  - avail(b) ·¬ant(b)
  - avail(b) · ant(b)
  - ¬avail(b) · ant(b)
- Finde Start des E-Pfad-Suffixes: Suche Block  $b_m$  mit  $\neg$ avail $(b_m) \cdot$  ant $(b_m) \cdot \sum_p$  avail(p)
- Suche rückwärts von b<sub>m</sub> durch Segmente avail(b) · ant(b) avail(b) · ¬ant(b) bis Start des E-Pfads b<sub>i</sub> erreicht
- Suche vorwärts von b<sub>m</sub> durch Segment
   ¬avail(b) · ant(b)
   bis Ende des E-Pfades erreicht



A. Koch

inleitung

. .

Nonzepte

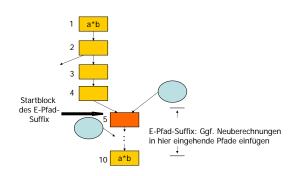
. . .

F-Pfade

atonfluce

### Aktionen nach Bestimmen des E-Pfades





nleitung

A Koch

Conzepte

Notation

E-Pfade

atenfluss

#### Start des E-Pfades Sichere Wert von e in $t_e$

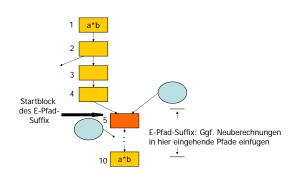
E-Pfad-Suffix Füge Neuberechnungen  $t_e := e$  in eingehende Pfade ein

Ende des E-Pfades Ersetzte redundante Berechnung von e durch  $t_e$ 



#### Aktionen nach Bestimmen des E-Pfades





A Koch

A. Kocn

inieitung

. .

. to...zopto

Linduini

101411011

E-Pfade

atenfluss

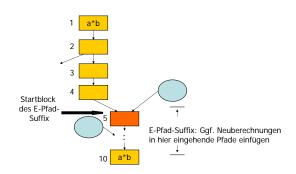
Start des E-Pfades Sichere Wert von e in  $t_e$  E-Pfad-Suffix Füge Neuberechnungen  $t_e:=e$  in eingehende Pfade ein

Ende des E-Pfades Ersetzte redundante Berechnung von e durch  $t_e$ 



#### Aktionen nach Bestimmen des E-Pfades





A Koch

A. Koch

inleitung

...

rtonzopio

......

E-Pfade

atenfluss

Start des E-Pfades Sichere Wert von e in  $t_e$ 

E-Pfad-Suffix Füge Neuberechnungen  $t_e := e$  in eingehende Pfade ein

Ende des E-Pfades Ersetzte redundante Berechnung von e durch  $t_e$ 





• Löse E-Pfad-Problem für alle Ausdrücke gleichzeitig

- Erinnerung: Ergebnisse der Prädikate sind Bitvektoren
- Bitweise Verknüpfung mit AND und OR
- Berechnet nicht nur E-Pfade
- Direkte Bestimmung von
  - Blöcken mit zu eliminierenden Ausdrücken
  - Einfügestellen (Blöcke, Kanten) für Neuberechnungen
  - Stellen f
    ür Sichern von Werten



A. Koch

inleitung

RE

Konzepte

Notation

E-Pfade



- Löse E-Pfad-Problem für alle Ausdrücke gleichzeitig
- Erinnerung: Ergebnisse der Prädikate sind Bitvektoren
- Bitweise Verknüpfung mit AND und OF
- Berechnet nicht nur E-Pfade
- Direkte Bestimmung vor
  - Blöcken mit zu eliminierenden Ausdrücken
  - Einfügestellen (Blöcke, Kanten) für Neuberechnungen
  - Stellen f
    ür Sichern von Werten



A. Koch

inleitung

KE

Konzepte

Notation

E-Pfade



- Löse E-Pfad-Problem für alle Ausdrücke gleichzeitig
- Erinnerung: Ergebnisse der Prädikate sind Bitvektoren
- Bitweise Verknüpfung mit AND und OR



A Koch

F-Pfade



- Löse E-Pfad-Problem für alle Ausdrücke gleichzeitig
- Erinnerung: Ergebnisse der Prädikate sind Bitvektoren
- Bitweise Verknüpfung mit AND und OR
- Berechnet nicht nur E-Pfade
- Direkte Bestimmung von

Blöcken mit zu eliminierenden Ausdrücken
Einfügestellen (Blöcke, Kanten) für Neuberechnungen
Stellen für Sichern von Werten

UC

A. Koch

inleitung

KE

Konzepte

Notation

E-Pfade



- Löse E-Pfad-Problem für alle Ausdrücke gleichzeitig
- Erinnerung: Ergebnisse der Prädikate sind Bitvektoren
- Bitweise Verknüpfung mit AND und OR
- Berechnet nicht nur E-Pfade
- Direkte Bestimmung von
  - Blöcken mit zu eliminierenden Ausdrücken
  - Einfügestellen (Blöcke, Kanten) für Neuberechnunger
  - Stellen für Sichern von Werten



A. Koch

inleitung

1112

vousebre

votation

E-Pfade

## Parallele Lösung der E-Pfad-Bestimmung



Löse E-Pfad-Problem für alle Ausdrücke gleichzeitig

- Erinnerung: Ergebnisse der Prädikate sind Bitvektoren
- Bitweise Verknüpfung mit AND und OR
- Berechnet nicht nur E-Pfade
- Direkte Bestimmung von
  - Blöcken mit zu eliminierenden Ausdrücken
  - Einfügestellen (Blöcke, Kanten) für Neuberechnunger
  - Stellen für Sichern von Werten



A. Koch

inleitung

RE

Conzepte

Lilaaiii

votation

E-Pfade

## Parallele Lösung der E-Pfad-Bestimmung



• Löse E-Pfad-Problem für alle Ausdrücke gleichzeitig

- Erinnerung: Ergebnisse der Prädikate sind Bitvektoren
- Bitweise Verknüpfung mit AND und OR
- Berechnet nicht nur E-Pfade
- Direkte Bestimmung von
  - Blöcken mit zu eliminierenden Ausdrücken
  - Einfügestellen (Blöcke, Kanten) für Neuberechnungen
  - Stellen für Sichern von Werten



A. Koch

inleitung

KE

Konzepte

NOLALIOIT

E-Pfade

# Parallele Lösung der E-Pfad-Bestimmung



Löse E-Pfad-Problem für alle Ausdrücke gleichzeitig

- Erinnerung: Ergebnisse der Prädikate sind Bitvektoren
- Bitweise Verknüpfung mit AND und OR
- Berechnet nicht nur E-Pfade
- Direkte Bestimmung von
  - Blöcken mit zu eliminierenden Ausdrücken
  - Einfügestellen (Blöcke, Kanten) für Neuberechnungen
  - Stellen f
    ür Sichern von Werten



A. Koch

inieitung

'KE

Konzepte

Linduini

votation

E-Pfade

Jatenfluss



# Datenflussproblem

OC

l. Koch

RE

Conzepte

E-Piad PRE

- Dr. I

E-Ptade

Datenfluss



- Datenflussproblem über Prädikate von Ausdrücken
- Zerlegung in
  - Verschiedene Teilprobleme
  - Vorberechenbare Eigenschaften
    - Herleitbare Eigenschaften (durch DF-Löser)

OC

A. KOCII

RE

Konzepte

.. .

Notation

E-Pfade



- Datenflussproblem über Prädikate von Ausdrücken
- Zerlegung in
  - Verschiedene Teilprobleme
  - Vorberechenbare Eigenschaften
  - Herleitbare Eigenschaften (durch DF-Löser)

OC

A. NOCH

DE

Konzepte

E-Plau PR

Notation

E-Pfade



- Datenflussproblem über Prädikate von Ausdrücken
- Zerlegung in
  - Verschiedene Teilprobleme
  - Vorberechenbare Eigenschaften
  - Herleitbare Eigenschaften (durch DF-Löser

00

A. Koch

inleitung

Konzepte

NOtation

E-Ptade



- Datenflussproblem über Prädikate von Ausdrücken
- Zerlegung in
  - Verschiedene Teilprobleme
  - Vorberechenbare Eigenschaften
  - Herleitbare Eigenschaften (durch DF-Löser)

OC

A. Koch

inieitung

Konzepte

Notation

E-Piade



- Datenflussproblem über Prädikate von Ausdrücken
- Zerlegung in
  - Verschiedene Teilprobleme
  - Vorberechenbare Eigenschaften
  - Herleitbare Eigenschaften (durch DF-Löser)

OC

A. Koch

inleitung

Konzepte

Notation

# Vorberechenbare Eigenschaften



#### Ausdruck ist lokal verfügbar (locally available)

comp(b) = TRUE: e wird in b berechnet (computed) und seine Operanden hinterher nicht überschrieben

 $ightharpoonup e \in \mathsf{DEExpr}(\mathsf{b})$ 

Ausdruck ist lokal vorziehbar (locally anticipatable)

antloc(b) = TRUE: Operanden von e werden vor Berechnung nicht zugewiesen

 $ightharpoonup e \in \mathsf{UEExpr}(\mathsf{b})$ 

Block ist transparent für Ausdruck

transp(b) = TRUE: b hat keine Zuweisungen an Operanden von e

 $\Rightarrow e \notin EXPRKILL(b)$ 

4□ ► 4₫ ► 4분 ► 4분 ► 900

OC

A. Koch

inieitung

Conzepte

E-FIAU FNI

Dfada

# Vorberechenbare Eigenschaften



#### Ausdruck ist lokal verfügbar (locally available)

comp(b) = TRUE: e wird in b berechnet (computed) und seine Operanden hinterher nicht überschrieben

 $ightharpoonup e \in \mathsf{DEExpr}(\mathsf{b})$ 

#### Ausdruck ist lokal vorziehbar (locally anticipatable)

antloc(b) = TRUE: Operanden von e werden vor Berechnung nicht zugewiesen

 $\Rightarrow e \in \mathsf{UEExpr}(\mathsf{b})$ 

#### Block ist transparent für Ausdruck

transp(b) = TRUE: b hat keine Zuweisungen an Operanden von e

⇒e ∉ ExprKill(b)

4□ > 4₫ > 4불 > 4불 > ½ 900

ОС

A. Koch

inleitung

onzepte

z-Piad PR

NOtation

z-Ptade

# Vorberechenbare Eigenschaften



#### Ausdruck ist lokal verfügbar (locally available)

comp(b) = TRUE: e wird in b berechnet (computed) und seine Operanden hinterher nicht überschrieben

 $ightharpoonup e \in \mathsf{DEExpr}(\mathsf{b})$ 

## Ausdruck ist lokal vorziehbar (locally anticipatable)

antloc(b) = TRUE: Operanden von e werden vor Berechnung nicht zugewiesen

 $\Rightarrow e \in \mathsf{UEExpr}(\mathsf{b})$ 

#### Block ist transparent für Ausdruck

transp(b) = TRUE: b hat keine Zuweisungen an Operanden von e

 $ightharpoonup e \notin ExprKill(b)$ 

4□ > 4₫ > 4 분 > 4 분 > 1 9 9 0

ОС

A. Koch

nieitung RE

Conzepte

Natation

E-Pfade

## Beispiel für vorberechenbare Eigenschaften



...

A. Koch

inleitung

/----

E-Pfad PR

Notation

Datenfluss

....

B1

Bit 1: a+b
Bit 2: c+d

Bit 3: e+f

Bit 4: a+e

Bit 5: a+r

Bit 6: x+y

comp(B1) =111100

antloc(B1)=011000 transp(B1)=010010 a : = 42

w := a + b

x := c + d

y := e + f

z := a + e

e := 23

 $nicht\ UEExpr/ANTloc,\ DEExpr$ 

UEExpr/ANTloc, DEExpr

UEExpr/ANTloc, nicht DEExpr

nicht UEExpr/ANTloc, nicht DEExpr

4□ > 4ⓓ > 4틸 > 4틸 > □ ● 9Q@



# availin(b) e ist verfügbar bei Eintritt in b

availout(b) e ist verfügbar bei Austritt aus b

#### Berechnung

```
availin(b) = \prod_p \text{availout}(p)
availout(b) = availin(b) · transp(b) +
```

OC

I. Koch

nleitung

RE

onzepte

-FIAU FNI

Diada



availin(b) e ist verfügbar bei Eintritt in b availout(b) e ist verfügbar bei Austritt aus b

#### Berechnung

```
availin(b) = \prod_p availout(p)
```

 $availout(b) = availin(b) \cdot transp(b) + comp(b)$ 

OC

A. Koch

inleitung

1112

onzepte.

latation

-Pfade



availin(b) e ist verfügbar bei Eintritt in b availout(b) e ist verfügbar bei Austritt aus b

#### Berechnung

```
availin(b) = \prod_p availout(p)
```

 $availout(b) = availin(b) \cdot transp(b) + comp(b)$ 

OC

A. Koch

inleitung

1112

onzepte.

latation

-Pfade



```
\operatorname{availin}(b) \ e ist verfügbar bei Eintritt in b \operatorname{availout}(b) \ e ist verfügbar bei Austritt aus b
```

## Berechnung

```
availin(b) = \prod_{p} availout(p)
```

 $availout(b) = availin(b) \cdot transp(b) + comp(b)$ 

OC

A. Koch

inieitung

onzepte

latation

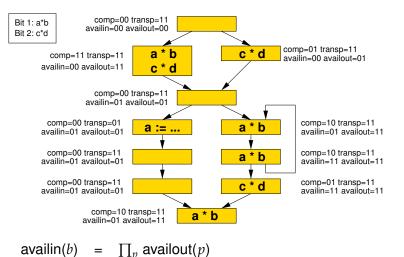
E-Pfade

## Beispiel verfügbare Ausdrücke

In Prädikatennotation

availout(b)





=  $availin(b) \cdot transp(b) + comp(b)$ 

OC

A. Koch

ınıeıtunç

Conzepte

E-Pfad PR

otation



# antin(b) e ist vorziehbar bei Eintritt in b

 $\mathsf{antout}(b)$  e ist vorziehbar bei Austritt aus t

#### Berechnung

```
antin(b) = antout(b) \cdot transp(b) + antloc(b)
```

ОС

I. Koch

nleitung

onzente

:-Ptad PRI

Dfodo



antin(b) e ist vorziehbar bei Eintritt in b antout(b) e ist vorziehbar bei Austritt aus b

#### Berechnung

```
antin(b) = antout(b) · transp(b) + antloc(b)
antout(b) = \Pi_c antin(s)
```

OC

A. Koch

nleitung

conzepte

latation

-Pfade



antin(b) e ist vorziehbar bei Eintritt in b antout(b) e ist vorziehbar bei Austritt aus b

#### Berechnung

```
antin(b) = antout(b) · transp(b) + antloc(b)
antout(b) = \Pi_c antin(s)
```

OC

A. Koch

nleitung

conzepte

latation

-Pfade



antin(b) e ist vorziehbar bei Eintritt in b antout(b) e ist vorziehbar bei Austritt aus b

## Berechnung

```
antin(b) = antout(b) \cdot transp(b) + antloc(b)
```

 $antout(b) = \prod_{s} antin(s)$ 

OC

A. Koch

inleitung

'onzonto

. בייי

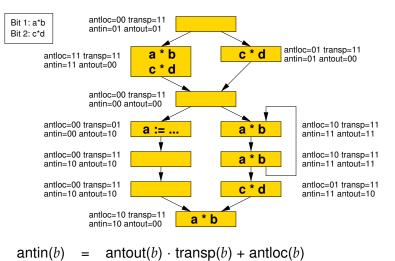
lotation

E-Pfade

## Beispiel vorziehbare Ausdrücke

In Prädikatennotation





OC

A. Koch

nleitung

Konzepte

ntation

-Pfade

Datenfluss

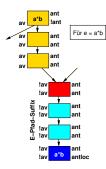
32/77



Datenfluss

epsin(b) Blockanfang von b liegt auf einem E-Pfad-Suffix für e

epsout(b) Blockende von b liegt auf einem E-Pfad-Suffix für e



#### Berechnung

```
\begin{array}{lll} \operatorname{epsin}(b) &=& (\sum_p (\operatorname{availout}(p) + \operatorname{epsout}(p))) \cdot \operatorname{antin}(b) \cdot \neg \operatorname{availin}(b) \\ \operatorname{epsout}(b) &=& \operatorname{epsin}(b) \cdot \neg \operatorname{antloc}(b) \end{array}
```

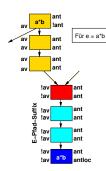




Datenfluss

epsin(b) Blockanfang von b liegt auf einem E-Pfad-Suffix für e

epsout(b) Blockende von b liegt auf einem E-Pfad-Suffix für e



#### Berechnung

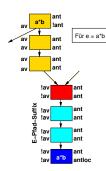
```
epsin(b) = (\sum_{p} (availout(p) + epsout(p))) \cdot antin(b) \cdot \neg availin(b)
epsout(b) = epsin(b) \cdot \neg antloc(b)
```



Datenfluss

epsin(b) Blockanfang von b liegt auf einem E-Pfad-Suffix für e

epsout(b) Blockende von b liegt auf einem E-Pfad-Suffix für e



#### Berechnung

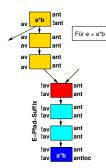
```
epsin(b) = (\sum_{p} (availout(p) + epsout(p))) \cdot antin(b) \cdot \neg availin(b)
epsout(b) = epsin(b) \cdot \neg antloc(b)
```



Datenfluss

epsin(b) Blockanfang von b liegt auf einem E-Pfad-Suffix für e

epsout(b) Blockende von b liegt auf einem E-Pfad-Suffix für e



## Berechnung

 $(\sum_{p} (availout(p) + epsout(p))) \cdot antin(b) \cdot \neg availin(b)$ epsin(b)

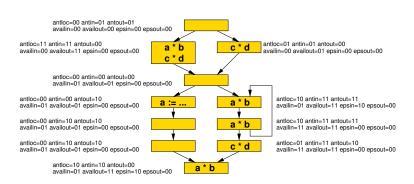
 $epsin(b) \cdot \neg antloc(b)$ epsout(b)



# Beispiel E-Pfad-Suffix



In Prädikatennotation



OC

A. NOCH

nleitung

onzepte

E-Pfad PRE

otation

E-Pfade

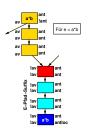
$$\begin{array}{lcl} \operatorname{epsin}(b) & = & (\sum_p (\operatorname{availout}(p) + \operatorname{epsout}(p))) \cdot \operatorname{antin}(b) \cdot \neg \operatorname{availin}(b) \\ \operatorname{epsout}(b) & = & \operatorname{epsin}(b) \cdot \neg \operatorname{antloc}(b) \end{array}$$

# Idee: Bestimmen der Blöcke der redundanten Berechnungen

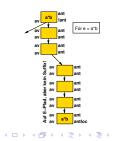


Redundante  $e \in \mathsf{UEExpr}(b_k)$  liegen im Endblock  $b_k$  eines E-Pfades

E-Pfad-Suffix  $\neq \emptyset$ Endblock des E-Pfad-Suffixes  $\Leftrightarrow$  epsin(b)  $\cdot$  antloc(



E-Pfad-Suffix =  $\emptyset$ Endblock des E-Pfades  $\Leftrightarrow$  availin(b)  $\cdot$  antloc(b)



OC

A. KOCH

RE

onzepte

E-Piad PRI

E-Pfade

# Idee: Bestimmen der Blöcke der redundanten Berechnungen

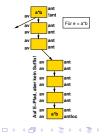


Redundante  $e \in \mathsf{UEExpr}(b_k)$  liegen im Endblock  $b_k$  eines E-Pfades

E-Pfad-Suffix  $\neq \emptyset$ Endblock des E-Pfad-Suffixes  $\Leftrightarrow$  epsin(b) · antloc(b)

E-Pfad-Suffix =  $\emptyset$ Endblock des E-Pfades

 $\Leftrightarrow$  availin(b) · antloc(b)





A. Koch

nleitung

Conzepte

:-Ptad PRE

-Pfade



redund(b) Berechnung von e in b ist redundant und kann durch  $t_e$  ersetzt werden

#### Berechnung

 $redund(b) = (epsin(b) + availin(b)) \cdot antloc(b)$ 

Genauer: Alle lokal antizipierbaren e, also  $e \in \mathsf{UEExpr}(b_k)$  sind redundant.

⇒Können eliminiert werden

UU

A. Koch

nleitung

KE

Conzepte

Notation

-Pfade



redund(b) Berechnung von e in b ist redundant und kann durch  $t_e$  ersetzt werden

#### Berechnung

 $redund(b) = (epsin(b) + availin(b)) \cdot antloc(b)$ 

Genauer: Alle lokal antizipierbaren e, also  $e \in \mathsf{UEExpr}(b_k)$  sind redundant.

⇒Können eliminiert werden



A. Koch

eitung

RE

Conzepte

lotation

-Pfade



redund(b) Berechnung von e in b ist redundant und kann durch  $t_e$  ersetzt werden

#### Berechnung

 $redund(b) = (epsin(b) + availin(b)) \cdot antloc(b)$ 

Genauer: Alle lokal antizipierbaren e, also  $e \in \mathsf{UEExpr}(b_k)$  sind redundant.

→Können eliminiert werden

UC

A. Koch

nleitung

Conzepte

i-Fidu Fni

-Pfade





redund(b) Berechnung von e in b ist redundant und kann durch  $t_e$  ersetzt werden

#### Berechnung

 $redund(b) = (epsin(b) + availin(b)) \cdot antloc(b)$ 

Genauer: Alle lokal antizipierbaren e, also  $e \in \mathsf{UEExpr}(b_k)$  sind redundant.

→Können eliminiert werden

UU

A. Koch

nleitung RF

Conzepte

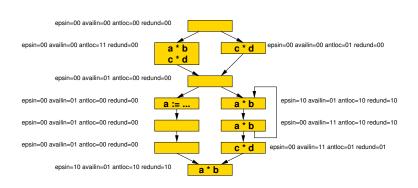
Jotation

-Pfade

# Beispiel Redundanz

In Prädikatennotation





OC A Koch

A. NOCH

Inleitung

Conzepte

-Pfad PRE

. ....

- 1 1000

Datenfluss

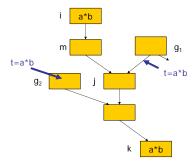
 $redund(b) = (epsin(b) + availin(b)) \cdot antloc(b)$ 

# Einfügestellen für neue Berechnungen

systems + applications

Wo muß  $t_e := e$  eingefügt werden?

#### An Eintrittspunkten in den Pfad!



- nleitung RE onzepte Pfad PRE
- -Pfade
- Datenfluss

- Wenn alle Nachfolger von externem Block in E-Pfad In Block
- Sonst in aufgeteilte Kante

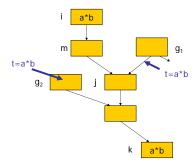


## Einfügestellen für neue Berechnungen

Wo muß  $t_e := e$  eingefügt werden?



#### An Eintrittspunkten in den Pfad!



- Wenn alle Nachfolger von externem Block in E-Pfad: In Block
- Sonst in aufgeteilte Kante

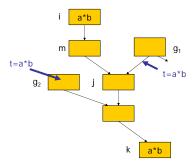


## Einfügestellen für neue Berechnungen

systems + applications

Wo muß  $t_e := e$  eingefügt werden?

#### An Eintrittspunkten in den Pfad!



- Wenn alle Nachfolger von externem Block in E-Pfad: In Block
- Sonst in aufgeteilte Kante



A. Koch

leitung

Conzepte

E-Pfad PR

Notation

-Pfade

# Einfügestellen für neue Berechnungen bestimmen



```
\begin{array}{ll} \mathsf{insert}(b_h) \ \mathsf{F\"{u}ge} \ \mathsf{t}_e \ := e \ \mathsf{am} \ \mathsf{Ende} \ \mathsf{von} \ \mathsf{Block} \ b_h \ \mathsf{ein} \\ \mathsf{insert}(b_h,b_j) \ \mathsf{F\"{u}ge} \ \mathsf{t}_e \ := e \ \mathsf{in} \ \mathsf{aufgeteilte} \ \mathsf{Kante} \ (b_h,b_j) \ \mathsf{ein} \\ b_h \notin \mathsf{E-Pfad}, \ b_j \in \mathsf{E-Pfad} \ (b_i,\ldots,b_k] \end{array}
```

#### Berechnung

```
\begin{array}{lll} \mathsf{insert}(b_h) & = & \neg \mathsf{availout}(b_h) \cdot \neg \; \mathsf{epsout}(b_h) \cdot \prod_s \; \mathsf{epsin}(s) \\ \mathsf{insert}(b_h, b_j) & = & \neg \mathsf{availout}(b_h) \cdot \neg \; \mathsf{epsout}(b_h) \cdot \neg \; \mathsf{insert}(b_h) \cdot \mathsf{epsin}(b_j) \end{array}
```

OC

I. Koch

nleitung

onzepte

-Piad PRE

F-Pfade

# Einfügestellen für neue Berechnungen bestimmen



```
\begin{array}{ll} \mathsf{insert}(b_h) \ \mathsf{F\"{u}ge} \ \mathsf{t}_e \ := e \ \mathsf{am} \ \mathsf{Ende} \ \mathsf{von} \ \mathsf{Block} \ b_h \ \mathsf{ein} \\ \mathsf{insert}(b_h,b_j) \ \mathsf{F\"{u}ge} \ \mathsf{t}_e \ := e \ \mathsf{in} \ \mathsf{aufgeteilte} \ \mathsf{Kante} \ (b_h,b_j) \ \mathsf{ein} \\ b_h \notin \mathsf{E-Pfad}, \ b_j \in \mathsf{E-Pfad} \ (b_i,\ldots,b_k] \end{array}
```

#### Berechnung

```
\begin{array}{lll} \mathsf{insert}(b_h) & = & \neg \mathsf{availout}(b_h) \cdot \neg \; \mathsf{epsout}(b_h) \cdot \prod_s \; \mathsf{epsin}(s) \\ \mathsf{insert}(b_h,b_j) & = & \neg \mathsf{availout}(b_h) \cdot \neg \; \mathsf{epsout}(b_h) \cdot \neg \; \mathsf{insert}(b_h) \cdot \mathsf{epsin}(b_j) \end{array}
```

oc

A. Koch

nleitung

Conzepte

-FIAU FNE

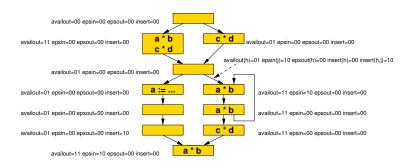
E-Pfade

#### Beispiel Einfügen von Berechnungen



OC

Datenfluss



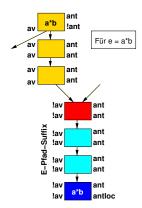
 $\text{insert}(b_h) = \neg \text{availout}(b_h) \cdot \neg \text{ epsout}(b_h) \cdot \prod_s \text{ epsin}(s)$   $\text{insert}(b_h, b_j) = \neg \text{availout}(b_h) \cdot \neg \text{ epsout}(b_h) \cdot \neg \text{ insert}(b_h) \cdot \text{epsin}(b_j)$ 



## Sichern von Berechnungsergebnissen



#### Am Anfang des E-Pfades!



A Koch

A. KOCII

one

Konzepte

E-Pfad PRI

- - - -

E-Pfade

Datenfluss

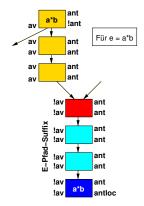
⇒Startblock des E-Pfades bestimmen!



## Sichern von Berechnungsergebnissen



#### Am Anfang des E-Pfades!



A. Koch

71. 110011

RE

Konzepte

E-Pfad PRI

---

E-Plade - ..

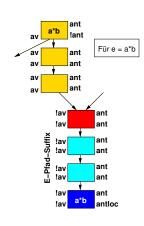
Datenfluss

Startblock des E-Pfades bestimmen!





- Beginne bei bekanntem Block und suche rückwärts
- Beginne bei Startblock des E-Pfad-Suffix
   ... falls E-Pfad einen
- Sonst: Suche von Endblock des E-Pfades aus rückwärts
- Bis nicht-redundante Berechnung von e gefunden



oc

A. Koch

inleitung

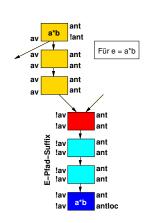
Konzepte

Intation

E-Pfade



- Beginne bei bekanntem Block und suche rückwärts
- Beginne bei Startblock des E-Pfad-Suffix
  - ... falls E-Pfad einen Suffix hat
- Sonst: Suche von Endblock des E-Pfades aus rückwärts
- Bis nicht-redundante Berechnung von e gefunden



OC

A. Koch

inleitung BE

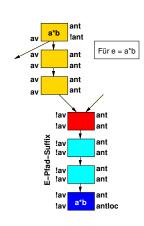
Konzepte

lotation

E-Pfade



- Beginne bei bekanntem Block und suche rückwärts
- Beginne bei Startblock des E-Pfad-Suffix
  - ... falls E-Pfad einen Suffix hat
- Sonst: Suche von Endblock des E-Pfades aus rückwärts
- Bis nicht-redundante Berechnung von e gefunden



OC

A. Koch

inleitung

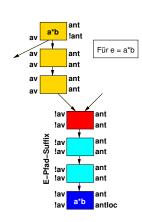
Konzepte

lotation

E-Pfade



- Beginne bei bekanntem Block und suche rückwärts
- Beginne bei Startblock des E-Pfad-Suffix
  - ... falls E-Pfad einen Suffix hat
- Sonst: Suche von Endblock des E-Pfades aus rückwärts
- Bis nicht-redundante Berechnung von e gefunden



OC

A. Koch

inleitung RE

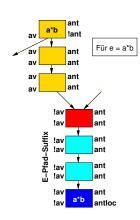
Konzepte

lotation

E-Pfade



- Beginne bei bekanntem Block und suche rückwärts
- Beginne bei Startblock des E-Pfad-Suffix
  - ... falls E-Pfad einen Suffix hat
- Sonst: Suche von Endblock des E-Pfades aus rückwärts
- Bis nicht-redundante Berechnung von e gefunden



ОС

A. Koch

inleitung RE

Konzepte

Votation

E-Pfade

Datenfluss

#### Berechnung des Startblocks



svupin(b) Sichere

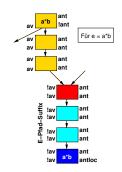
Berechnungsergebnis für e über Blockeingang von b

svupout(b) Sichere

Berechnungsergebnis für *e* über Blockausgang von *b* 

save(b) Sichere

Berechnungsergebnis in  $t_e$ in Block b



```
ОС
```

A. Koch

RE

Konzepte

Notation

E-Pfade

Datenfluss

#### Berechnung

```
svupout(b) = (\sum_{s} (epsin(s) + redund(s) + svupin(s))) \cdot availout(b)
svupin(b) = svupout(b) \cdot \neg comp(b)
save(b) = svupout(b) \cdot \neg comp(b) \cdot \neg (redund(b) \cdot transp(b))
```

#### Berechnung des Startblocks



Datenfluss

svupin(b) Sichere

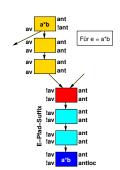
Berechnungsergebnis für e über Blockeingang von b

svupout(b) Sichere

Berechnungsergebnis für e über Blockausgang von b

save(b) Sichere Berechnungsergebnis in t<sub>e</sub>

in Block h



#### Berechnung

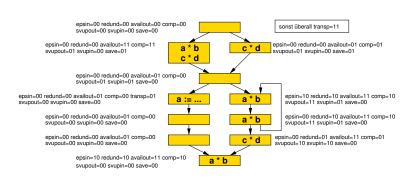
save(b)

```
(\sum_{s} (epsin(s) + redund(s) + svupin(s))) \cdot availout(b)
svupout(b)
 svupin(b)
                      svupout(b) \cdot \neg comp(b)
```

 $svupout(b) \cdot comp(b) \cdot \neg (redund(b) \cdot transp(b))$ 

## Beispiel Bestimmung des E-Pfad-Startblöcke Stellen zum Sichern der Berechungsergebnisse

embedded systems + applications



 $save(b) = svupout(b) \cdot comp(b) \cdot \neg (redund(b) \cdot transp(b))$ 

OC

A. Koch

leitung

. . . . . . . . . . . .

-. . . - - -

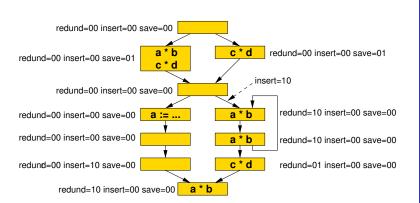
Notation

E-Pfade

## Beispiel Zusammenfassung Datenflussanalyse

Bisher berechnete Ergebnisse





oc

A. Koch

inleitung RF

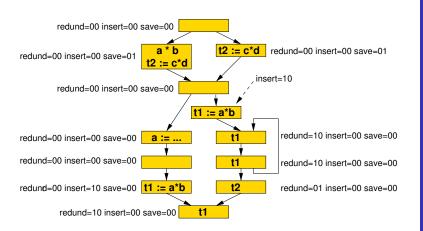
Conzepte

:-Piad PRI

D( )

## Beispiel PRE-Optimierung Auf Basis der Datenfluss-Ergebnisse





OC

A. Koch

inleitung

Conzepte

E-Pfad PRI

lotation

E-Ptade



PRE ist eine sehr m\u00e4chtige Optimierung

- Lösbar durch komplexes Datenflußproblen
  - Hier aber schon deutlich einfacher als klassische Verfahren!
- Noch weitere Verfeinerung möglich
  - Optimierung auf SSA-Form → SSA-PRE
  - Verfeinerte Stellung des Datenflussproblems
    - → D. Kumar 2006, geringerer Rechenaufwand



A. Kocn

inleitung

..\_

Konzepte

. I IQUI IIL

= Dfodo



- PRE ist eine sehr m\u00e4chtige Optimierung
- Lösbar durch komplexes Datenflußproblem
  - Hier aber schon deutlich einfacher als klassische Verfahren!
- Noch weitere Verfeinerung möglich
  - Optimierung auf SSA-Form → SSA-PRE
  - Verfeinerte Stellung des Datenflussproblems
    - → D. Kumar 2006, geringerer Rechenaufwand



A. Koch

inieitung

..\_

Conzepte

. . .

Dfodo



- PRE ist eine sehr m\u00e4chtige Optimierung
- Lösbar durch komplexes Datenflußproblem
  - Hier aber schon deutlich einfacher als klassische Verfahren!
- Noch weitere Verfeinerung möglich
  - Optimierung auf SSA-Form → SSA-PRE
     Verfeinerte Stellung des Datenflussproblems
     → D. Kumar 2006, geringerer Rechenaufwand



A. Koch

inleitung

/----

vorizebie

lotation

E-Pfade



- PRE ist eine sehr m\u00e4chtige Optimierung
- Lösbar durch komplexes Datenflußproblem
  - Hier aber schon deutlich einfacher als klassische Verfahren!
- Noch weitere Verfeinerung möglich
  - ullet Optimierung auf SSA-Form o SSA-PRE
  - Verfeinerte Stellung des Datenflussproblems
     → D. Kumar 2006, geringerer Rechenaufwand

oc

A. Koch

inleitung

/annanta

Conzepte

otation

E-Pfade



- PRE ist eine sehr m\u00e4chtige Optimierung
- Lösbar durch komplexes Datenflußproblem
  - Hier aber schon deutlich einfacher als klassische Verfahren!
- Noch weitere Verfeinerung möglich
  - $\bullet \ \, \text{Optimierung auf SSA-Form} \to \text{SSA-PRE} \\$
  - Verfeinerte Stellung des Datenflussproblems
     → D. Kumar 2006, geringerer Rechenaufwand



A. Koch

---

IZ-----

-Pfad PRF

otation

E-Pfade



- PRE ist eine sehr m\u00e4chtige Optimierung
- Lösbar durch komplexes Datenflußproblem
  - Hier aber schon deutlich einfacher als klassische Verfahren!
- Noch weitere Verfeinerung möglich
  - ullet Optimierung auf SSA-Form o SSA-PRE
  - Verfeinerte Stellung des Datenflussproblems
    - $\rightarrow$  D. Kumar 2006, geringerer Rechenaufwand



A. Koch

inleitung

Konzente

F-Pfad PRF

lotation

E-Ptade

Datenfluss