2017-04-25

Abhängigkeitsanalyse

Begriffe bei Graphen

- schwach zusammenhängend: von jedem Knoten kommt man zu jedem anderen Knoten
- stark zusammenhängend: wie schwach, nur unter Berücksichtigung der Kantenrichtung
- einfach zusammenhängend: nur eine Kante für Zusammenhang
- mehrfach zusammenhängend: mehrere Kanten für Zusammenhang
- Starke/schwache Zusammenhangskomponenten: Teilgraphen mit starkem/schwachem Zusammenhang
- azyklische Kondensation: Kanten zwischen den starken Zusammenhangskomponenten und diese als Superknoten

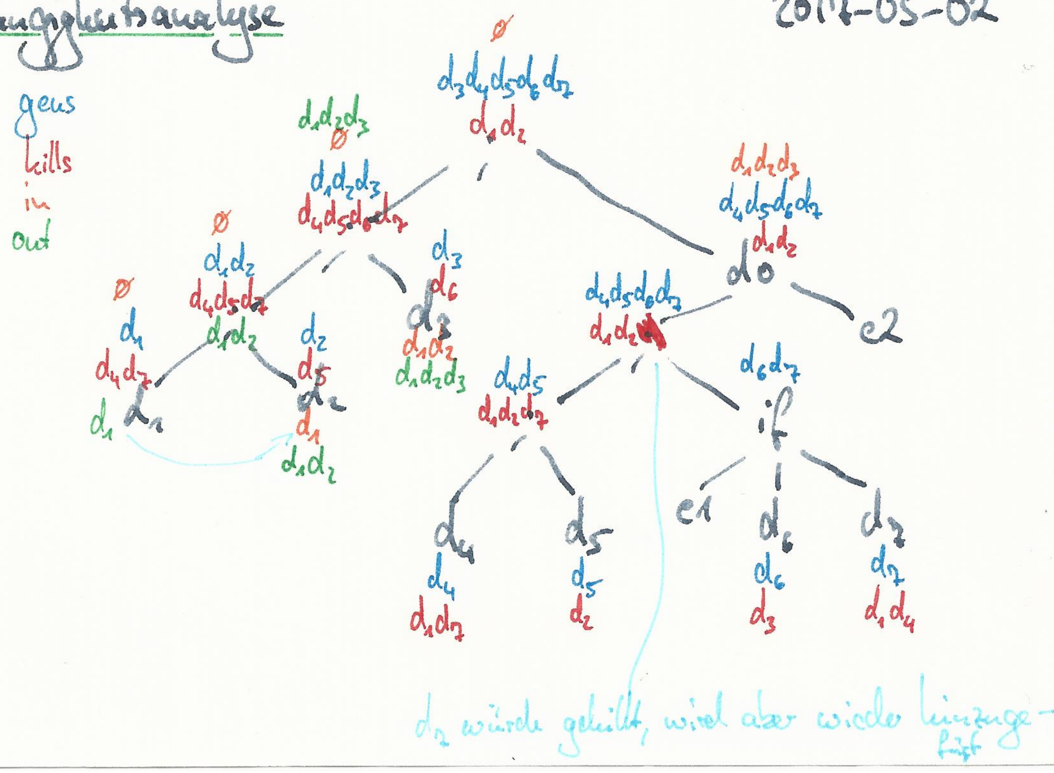
Traditionelle Abhängigkeitsanalyse

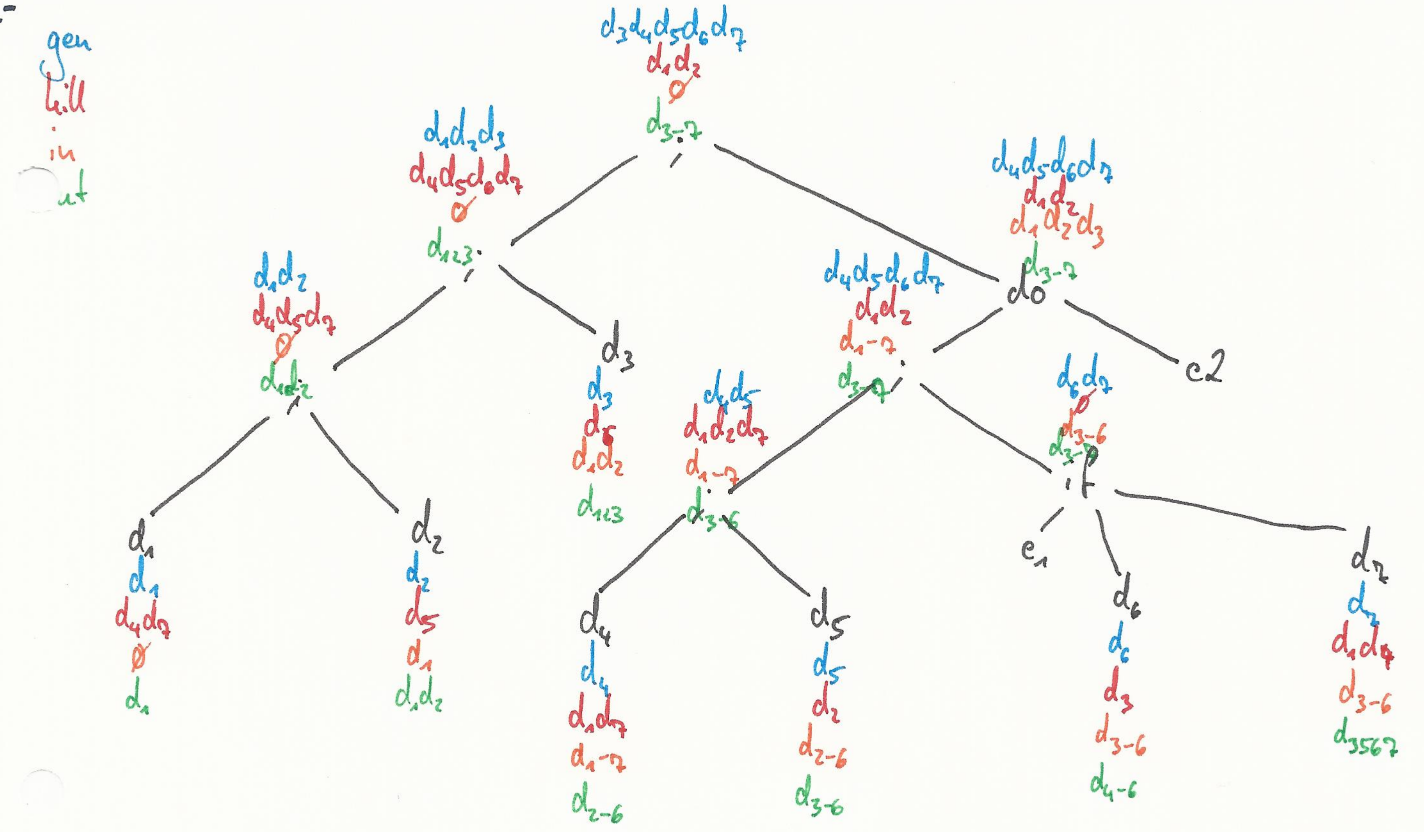
2017-05-02

$D_i = \{d_1, d_4, d_7\}$

$D_j = \{d_2, d_5\}$

$D_a = \{d_3, d_6\}$





Dieses Verfahren unterstützt so keine Arrays. Um diese zu modellieren, hat man dann so viele Variablen ~~mit~~ wie Arrayzellen. Allerdings sind die Mengenoperationen teuer → skaliert nicht.

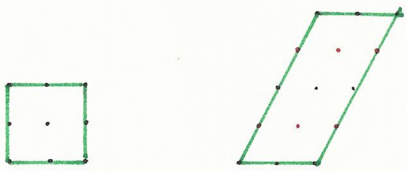
2017-05-09

Das Polyedermodell

Der Schnitt von endlich vielen Halbräumen ist ein Polyeder. Ein beschränktes Polyeder heißt Polytop. Durch Schnitt von Hyperebenen durch Räume entstehen Halbräume. Hyperebenen sind $n-1$-dimensionale Ebenen im n-dimensionalen Raum.

Mathematische Grundlagen

2017-05-16



nicht unimodular

Gauss-Elimination für Ganzzahlige Gleichungssysteme (Diophantische GS)

- Zeilen vertauschen
- Vielfaches einer Zeile zu anderer addieren
- Multiplikation einer Zeile mit ~~bel. Skalar~~ -1, um Ganzzahligkeit zu erhalten (für unimodulare Transformation)

<u>Beispiel:</u> Zeilen-Stufenform

$$\begin{array}{c} \quad A \\ \left(\begin{array}{ccc|ccc} 4 & 4 & 1 & 1 & 0 & 0 \\ 6 & 0 & 1 & 0 & 1 & 0 \\ 4 & 3 & 2 & 0 & 0 & 1 \end{array}\right) \end{array} \qquad \begin{array}{l} \text{Tausch } I-II \\ (-1)\cdot II \xrightarrow{+} III \\ \rightsquigarrow \end{array}$$

$$\left(\begin{array}{ccc|ccc} 6 & 0 & 1 & 0 & 1 & 0 \\ 4 & 4 & 1 & 1 & 0 & 0 \\ 0 & -1 & 1 & -1 & 0 & 1 \end{array}\right) \qquad \begin{array}{l} (-1) II \xrightarrow{+} I \\ \rightsquigarrow \end{array}$$

$$\left(\begin{array}{ccc|ccc} 2 & -4 & 0 & -1 & 1 & 0 \\ 4 & 4 & 1 & 1 & 0 & 0 \\ 0 & -1 & 1 & -1 & 0 & 1 \end{array}\right) \qquad \begin{array}{l} (-2) I \xrightarrow{+} II \\ \rightsquigarrow \end{array}$$

$$\left(\begin{array}{ccc|ccc} 2 & -4 & 0 & -1 & 1 & 0 \\ 0 & 12 & 1 & 3 & -2 & 0 \\ 0 & -1 & 1 & -1 & 0 & 1 \end{array}\right) \qquad \rightsquigarrow$$

$$\begin{array}{c} S \qquad\qquad\qquad U \\ \left(\begin{array}{ccc|ccc} 2 & -4 & 0 & -1 & 1 & 0 \\ 0 & 12 & 1 & 3 & -2 & 0 \\ 0 & 0 & 13 & -9 & -2 & 12 \end{array}\right) \end{array} \qquad U \cdot A = S$$

Bsp.: $4x_1 + 6x_2 = 8$

Matrixform: $(4 \;\; 6 \mid 8)$

Gauß: x_2 beliebig, $x_1 = \frac{4-3x_2}{2}$

$ggT(4,6) = 2$, teilt $8 \Rightarrow$ lösbar

$$\underset{A}{\begin{pmatrix} 4 \\ 6 \end{pmatrix}} \left| \begin{matrix} 1 & 0 \\ 0 & 1 \end{matrix} \right) \rightsquigarrow^{*} \underset{S}{\begin{pmatrix} 2 \\ 0 \end{pmatrix}} \left| \underset{U}{\begin{matrix} -1 & 1 \\ 3 & -2 \end{matrix}} \right)$$

„Kochrezept": zu lösen $x \cdot A = c$

$$\underbrace{x \cdot U^{-1}}_{t} \cdot \underbrace{U \cdot A}_{S} = c$$

$$t \cdot S = c$$

(1) $(A \mid E) \rightsquigarrow^{*} (S \mid U)$ E: Einheitsmatrix

(2) Lösung für t

(3) $t = x \cdot U^{-1} \quad | \cdot U$

$t \cdot U = x$

Aufgabe $(x_1 \;\; x_2)\begin{pmatrix} 4 \\ 6 \end{pmatrix} = 8$

$(t_1 \;\; t_2)\begin{pmatrix} 2 \\ 0 \end{pmatrix} = 8 \iff t_2$ beliebig, $t_1 = 4$

$\mathcal{L}_t = \{(4, t_2) : t_2 \in \mathbb{Z}\}$

$x = t \cdot U = (4 \;\; t_2) \cdot \begin{pmatrix} -1 & 1 \\ 3 & -2 \end{pmatrix} = (-4+3t_2, 4-2t_2),\; t_2 \in \mathbb{Z}$

$\mathbb{L}_x = \{(-4+3t_2, 4-2t_2) : t_2 \in \mathbb{Z}\}$

Beispiel:
$$\begin{aligned} 4x_1 + 6x_2 &= 8 \\ 15x_1 + 21x_2 + 6x_3 &= 9 \end{aligned}$$

alt: $x_1 \qquad + 6x_3 = 9$

Schema: $(x_1, x_2, x_3) A = (8, 9 \;\; 9)$

(1) $$\left(\begin{array}{ccc|ccc} 4 & 15 & 1 & 1 & 0 & 0 \\ 6 & 21 & 0 & 0 & 1 & 0 \\ 0 & 6 & 6 & 0 & 0 & 1 \end{array}\right) \rightsquigarrow^{*} \left(\begin{array}{ccc|ccc} 2 & 6 & -1 & -1 & 1 & 0 \\ 0 & 3 & 3 & 3 & -2 & 0 \\ 0 & 0 & 0 & -6 & 4 & 1 \end{array}\right)$$

A $\qquad$ S $\qquad$ U

(2) $(t_1, t_2, t_3) \cdot S = (8, 9\ 9)$

$\Leftrightarrow t_1 = 4, \quad t_2 = -5, \quad t_3 \in \mathbb{Z} \Rightarrow \vec{t} = (4, -5, t_3)$

$t_1 = 4, \quad t_2 = -t_1 + 3t_2 = 9 \Leftrightarrow t_2 = \frac{13}{3} \notin \mathbb{Z} \Rightarrow \mathbb{L} = \{\}$

$\Rightarrow x = (x_1, x_2, x_3) = (4, -5, t_3) \cdot U = (-19 - 6t_3, 4t_3, t_3)$

Kriterien für Abhängigkeiten

2017-05-23

- Programm mit Instanzen von Statements, die voneinander abhängig sind.

0. Identifikation der Teile, die voneinander abhängig sind (= Statement-Instanzen)
1. Speicherkonflikt → Konfliktgleichungssystem (Anzahl Arraydimensionen ist Anzahl Gleichungen; Anzahl Variablen im GS ist 2 · Anzahl Variablen im Programm)
2. Existenzproblem der Zugriffe ⇒ Existenz-Ungleichungssystem
3. Ordnung: sequentielle Ausführungsordnung auf den Statementinstanzen → lexikographische Ordnung: erst Indizes[1], dann textuelle Reihenfolge der Programmstatements.
4. Optimierung (optional): Entferne transitive Abhängigkeiten

[1] der gemeinsamen umgebenden Schleifen

S. Bernstein-Zusatz: mind. einer schreibt [optional?]

Bsp.:

$i = 0..n$
S: $A[i]$
T: $A[i-1]$

$S[i] \to T[i+1]$

$i = 0..n$
$A[2i]$
$B[i]$

keine Speicherkonflikte

$i = 0..n$
$A[2i]$
$A[2i+1]$

keine Speicherkonflikte

$i = 0..n-1$
$A[i]$
$A[2n-i]$

keine Speicherkonflikte, da es keine Existenz von Zugriffen gibt

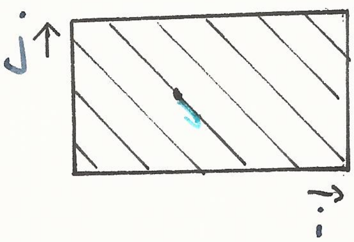
$i = 0..n$
$A[i] = \ldots A[i]$

selbe Statement-Instanz, daher keine Ausführungsordnung

$i = 0..n$
S: A
T: A

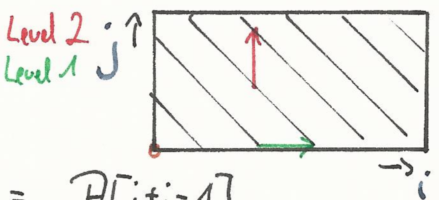
ohne Optimierung: alles, was nicht < (3) entspricht
mit Opt.: $S[i] \to T[i]$, $T[i] \to S[i+1]$

$i = 0..n$
$j = 0..n$
S: $A[i+j]$
T: $A[i+j]$



mit Opt.:
$S[i,j] \to T[i,j]$
$T[i,j] \to S[i+1, j-1]$

$i = 0..$
$j = 0..$
S: $A[i+j] = \ldots A[i+j-1]$



if $j > 0$ then
$S[i, j-1] \to S[i,j]$
else if $i > 0$ then
$S[i-1, 0] \to S[i, 0]$
else (1)

```
src[S[i,j]) =
    if j>0 then S[i,j-1]
    else if i>0 then S[i-1,0]
    else ⊥
```

Begriffe

Abh.: von Quelle zu Ziel

Iterationsvektor (Ziel) - Iterationsvektor (Quelle) lexikogr. nicht negativ

Wenn die Differenz 0 ist spricht man von schleifenunabhängiger oder textgetragener Abhängigkeit. Wenn Differenz konstant: uniforme Abhängigkeit → Distanzvektor ↳ Abhängigkeitsvektor

Ansonsten: Vergröberung: nur die Vorzeichen der Komponenten der Differenz → Richtungsvektor

Noch gröber: Level - die die Tiefe der tragenden Schleife

→ Position des ersten Nicht-Null-Eintrags im Richtungsvektor

Verallgemeinerte Richtungsvektoren

$$(1,1) \cup (1,0) \cup (1,-1) \cong (1,*)$$

Ziel \ Quelle	liest	schreibt
liest	input	true
schreibt	anti	output

flow = true + Optimierung

2017-05-30

for $i = 10$ to 200
  for $j = 7$ to 167
S: $X[2i+3, 5j-1, j] = \ldots$
T: $\ldots = X[i-1, 2i-6, 3j+2]$

(1) Speicherkonflikt:

$$2i+3 = i'-1$$
$$5j-1 = 2i'-6$$
$$j = 3j'+2$$

Banerjee notiert als Matrix:

$$\left.\begin{matrix} A = \begin{pmatrix} 2 & 0 & 0 \\ 0 & 5 & 1 \end{pmatrix} \\ a_0 = (3 \ \ {-1} \ \ 0) \end{matrix}\right\} \text{für } S$$

$$\left.\begin{matrix} B = \begin{pmatrix} 1 & 2 & 0 \\ 0 & 0 & 3 \end{pmatrix} \\ b_0 = (-1 \ \ {-6} \ \ 2) \end{matrix}\right\} \text{für } T$$

$$(i,j)\cdot A + a_0 = (i',j')\cdot B + b_0$$

$$\Leftrightarrow (i,j,i',j')\cdot\begin{pmatrix} A \\ -B \end{pmatrix} = b_0 - a_0$$

im Beispiel:

$$(i,j,i',j')\cdot\begin{pmatrix} 2 & 0 & 0 \\ 0 & 5 & 1 \\ 1 & 2 & 0 \\ 0 & 0 & 3 \end{pmatrix} = (-4, -5, 2)$$

$$\left(\begin{array}{ccc|cccc} 2 & 0 & 0 & 1 & & & 0 \\ 0 & 5 & 1 & & 1 & & \\ 1 & 2 & 0 & & & 1 & \\ 0 & 0 & 3 & 0 & & & 1 \end{array}\right) \rightsquigarrow^* \left(\begin{array}{ccc|cccc} -1 & -2 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 2 & 0 \\ 0 & 0 & 1 & 5 & 4 & 10 & 1 \\ 0 & 0 & 0 & 15 & 12 & 30 & 4 \end{array}\right)$$

(linker Block: S, rechter Block: U)

$$(t_1 \ t_2 \ t_3 \ t_4)\cdot S = (-4, -5, 2)$$

$$t_1 = 4, \quad t_2 = 3, \quad t_3 = -1, \quad t_4 \in \mathbb{Z}$$

$$(i,j,i',j') = (4,3,-1,t_4)\cdot U = (-2+15t_4, -1+12t_4, 30t_4, -1+4t_4)$$

(1) $\rightarrow S[-2+15t_4, -1+12t_4] \leftrightarrow T[30t_4, -1+4t_4]$ für alle $t_4 \in \mathbb{Z}$

$i = 15t_4 - 2 \Leftrightarrow t_4 = \frac{i+2}{15}$ für $15 \mid (i+2)$, d.h. nicht $\forall i \in$ Indexraum existiert die Abhängigkeit

(2) Existenz-Ungleichungssystem:

$$p_0 \le (i,j)\cdot P, \quad (i,j)\cdot Q \le q$$

im Beispiel: $P = Q = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$, $p_0 = (10, 7)$, $q_0 = (200, 167)$

$$\underbrace{\begin{matrix} 10 \le i\text{-}2 + 15t_4 \le 200 \\ 7 \le j\text{-}1 + 12t_4 \le 167 \end{matrix}}_{\text{Existenz von } (i,j)} \qquad \underbrace{\begin{matrix} 10 \le 30t_4 \le 200 \\ 7 \le -1 + 4t_4 \le 167 \end{matrix}}_{\text{Existenz von } (i',j')}$$

$\overset{FM}{\Rightarrow} \; 2 \le t_4 \le 6$

aktuelles Zwischenergebnis: (1) mit $t_4 \in \{2, 3, 4, 5, 6\}$

(3) Ordnung (für die Richtung der Abhängigkeit): Abh $S \to T \Leftrightarrow$

$$(i,j) \prec (i',j') \Leftrightarrow i < i' \vee (i = i' \wedge j < j') \vee (i = i' \wedge j = j' \wedge \underbrace{S \text{ vor } T}_{\text{true}})$$

$\Leftrightarrow i < i' \vee i = i' \wedge j \le j'$

$\Leftrightarrow -2 + 15t_4 < 30t_4 \vee \dots$

$\Leftrightarrow t_4 > -\frac{2}{15} \vee \dots \Leftrightarrow$ true für $t_4 \in \{2, 3, 4, 5, 6\}$

d.h. Abh. $S[-2+15t_4, -1+12t_4] \to T[30t_4, -1+4t_4]$

d.h. S ist Quelle und T ist Ziel

d.h. wg. S schreibt und T liest: Typ ist true dependency

(4) Banerjee macht keine Optimierung

Distanzvektor: $(30t_4, -1+4t_4) - (-2+15t_4, -1+12t_4) = (\underbrace{15t_4 + 2}_{>0 \; \forall t_4 \in \{2,..,6\}}, -8t_4)$

$\Rightarrow$ Level 1

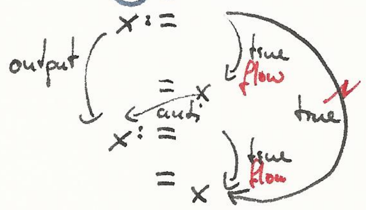
Distanzvektoren: $d_1 = (32, -16)$, $d_2 = (47, -24)$ ...

Statement-Dependence-Graph

$$S \xrightarrow{(15t_4+2,\ -8t_4)} T$$

2017-06-13

Abhängigkeitsanalyse nach Feautrier



Frage: Wer schrieb das „hier" gelesene Datum?

$\Rightarrow$ Optimierung nötig

Lockerung: Verzicht auf perfekte Verschachtelung

Bsp.:

$$
\begin{array}{ll}
 & \text{i = 0, n} \\
R: & \quad A[0,0] = \ldots \\
S: & \quad X[i] = A[i,0] \\
 & \quad \text{j = 1, n} \\
 & \qquad \text{k = 1, n} \\
T: & \qquad\quad A[i+k-1,\ j-1] = \ldots
\end{array}
$$

S-T-Konflikt:

(0) Instanzen: $S[i]$, $T[i', j', k']$

(1) Konfliktgleichungssystem:

$$i = i' + k' - 1$$
$$0 = j' - 1$$

(2) Existenzungleichungssystem:

$$0 \le i \le n \quad \leftarrow \text{Kontext}$$

$$\left.\begin{array}{l} 0 \le i' \le n \\ 1 \le j' \le n \\ 1 \le k' \le n \end{array}\right\} \text{Beschränkung für die Variablen}$$

(3) Ordnung: ~~S → T~~ T → S, d.h.

$$T[i', j', k'] \prec S[i]$$

$\Leftrightarrow i' < i \lor i' = i \land T$ textuell vor S $\Leftrightarrow i' < i \Leftrightarrow i' \le i - 1$

Resultat nach A_1: Reihenfolge (von innen nach außen):

k', j', i', i, n

(4) Optimierung:

$$i' \leq \min(n, i-1) = i-1 \quad \Rightarrow \quad i'_{max} = i-1$$

$$i' \geq \max(0, i-n+1)$$

$i-1 \geq 0 \Leftrightarrow i \geq 1$

$i-1 \geq i-n+1 \Leftrightarrow n \geq 2$

$$\left.\begin{array}{l} j' \geq 1 \\ j' \leq \min(n, 1) \end{array}\right\} j' = 1 = j'_{max}$$

$$k' \geq \max(-i'+i+1, 1) \Rightarrow k'_{max} \geq \max(2,1) = 2$$

$$k' \leq \min(-i'+i+1, n) \Rightarrow k'_{max} \leq \min(2, n) = 2$$

$$\Rightarrow k'_{max} = 2$$

$$\Rightarrow T[i-1, 1, 2] \rightarrow S[i] \text{ für } i \geq 1, n \geq 2, \underbrace{i \leq n}_{\text{Kontext}}$$

S-R-Konflikt:

(0) Instanzen: $S[i], R[i']$

(1) Konfliktgleichungssystem

$$0 = i'$$
$$0 = 0$$

(2) Existenzungleichungssystem:

$$0 \leq i' \leq n, \quad \underbrace{0 \leq i \leq n}_{\text{Kontext}}$$

(3) Ordnung: $R \rightarrow S$, d.h. $i' \leq i$ (R textuell vor S)

(4) Optimierung: $i' = i$

$i' = 0$ (wg. (2))

$$\Rightarrow R[0] \rightarrow S[0] \text{ bzw. } R[i] \rightarrow S[i], i = 0$$

Letzter Schritt: „Mischen" der potentiellen Quellen (mit lexikographischer Optimierung)

$$src(S[i]) = \begin{cases} R[i] & , i = 0 \\ T[i-1, 1, 2] & , i \neq 0 \end{cases}$$

⚠ Zufall: Bereiche disjunkt

Für den T-Fall in den Quellen werden alle Bedingungen von oben benötigt. Alle übrigen Fälle resultieren in $\perp$, was heißt, dass wir einen Wert gelesen haben, der vorher nie beschrieben worden ist.

Anti-Dependencies werden von diesem Verfahren nicht gefunden, was aber auch nicht nötig ist:

```
... = x          = x
x = ...          = x
x = ...        x =
```

orange wird gebraucht, außer ich berücksichtige die input auch

Schluss: Anti-Dependencies nicht mit Feautrier behandeln, da das Verfahren optimiert.

Funktionale Welt: Nur true-Deps (alles schön)

Idee: Überschreiben ist nicht nötig (s. funktionale Programmierung)

→ keine Output-, keine Anti-Dependencies

→ mehr Parallelität

Preis: Speicherverbrauch (Datenlokalität)

Idee: Single-Assignment-Form

Methode: Single-Assignment-Konversion

(1) Jeder Write in ein Statement-Array vollindiziert

(2) Beim Lesen Datenfluss lt. Feautrier-Analyse einsetzen

am Beispiel:

```
    [ i=0,n
R   |   R[i] = ...
S   |   X[i] = (i==0 ? R[i] : T[i-1,1,2]);
    |  [ j=1,n
    |  |  [ k=1,n
T   |  |  |   T[i,j,k] = ...
```

Heuristiken zur Abhängigkeitsanalyse

GCD-Test

$$4x_1 + 6x_2 = 8$$

$$\gcd(4,6) \mid 8$$

$$2 \mid 8$$

Teilt der gcd der Koeffizienten den konstanten Teil? Falls ja, ist das Gleichungssystem prinzipiell lösbar ist. Falls der gcd-Test nicht positiv ist, existieren definitiv keine Abhängigkeiten.

Um Abhängigkeiten zu finden, benötigt man auch das Existenzungleichungssystem.

Für Gleichungssystem: $xA = c$

$A \rightsquigarrow S$

$tS = c$ (wie Banerjee)

Falls es keine t gibt, existieren keine Abhängigkeiten.

Extremwerttest

```
for i=0..9
    ... = A[n+i]
    A[i] = ...
```

für Schreibzugriff			für Lesezugriff		
lower	upper	elim	lower	upper	elim
i	i	i	$n+i$	$n+i$	i
0	9		n	$n+9$	n
			1	∞	

Bei Überlappung der Intervalle könnte eine Abhängigkeit bestehen. Keine Abhängigkeiten bei disjunkten Mengen.

$i = h + i'$

$i - h - i' = 0$

$0 \leq i \leq 9$

$0 \leq i' \leq 9$

$h \geq 1$

lower	upper	elim
$i-h-i'$	$i-h-i'$	i
$-h-i'$	$9-h-i'$	i'
$-h-9$	$9-h$	h
$-\infty$	9	

$0 \overset{?}{\in}]-\infty; 9]$

$i < i'$

$0 \leq i$
$i \leq 9$
$0 \leq i'$
$i' \leq 9$
$i \leq i'-1$
$1 \leq h$

lower	upper	elim
$i-h-i'$	$i-h-i'$	i
$-h-i'$	$i'-1-h-i'$	i'
$-h-9$	$-1-h$	h
$-\infty$	-2	

$0 \overset{?}{\in}]-\infty; -2]$

↳ Konfliktgleichungssystem mit diesen Konstraints nicht lösbar, folglich kann es keinen Datenfluss (flow deps) geben.

Normalisierung der Arrayindizes

~~x = 1;~~
for i = 0..n :
  ~~x = x+2;~~ ~~x = 2i+3~~
  ~~A[x] = ...;~~ A[2i+3] = ...;
  ... = A[i];
}
x = 2n+3;

x = 1;
for i = ~~0~~1..n : A[1] = 3;
  ~~x = 2(i-1)+5;~~ x = 2i+3;
  A[x] = ...;
  x ~~= 2i+5;~~
x = 2n+5;

$x = \ldots$
$\quad = x$
$\quad = x$
$x \cancel{= \ldots}$
$\quad = \ldots x$

Inlining → Parallelisieren → Common Subexpression Elimination

~~Ska~~ Scalar Expansion

for in in range():
    $A[i] = \ldots$
    $\ldots = A[i] \cdot A[i+1];$

→

for i in range():
    $A'[i] =$
for i in range():
    $\cancel{A[i] = A'[i]}$
    $\ldots = \cancel{A[i]} \cdot A'[i-1]$

Verfahren via ~~Q~~ FADA

2017-06-27

Kontext: Read-Stmt: 7

Operations: $7[i,j] : 0 \le i,j \le N$

Annotation: Stmt 3: $3[i',j'] : 0 \le j', i' \le N \wedge i+j = i'+j'+1 \leftarrow W(3)$

Stmt 6: $6[i',j'] : 0 \le i',j' \le N \wedge P(i',j') \quad \leftarrow W(6)$

Durchlauf: Ordnung: $i = i', j = j'$

$src^{(0)} = \{\perp\}$

4 hoch: $src_4^{(1)} = \{\perp\}$

6 hoch: $src_6^{(1)} = src^{(0)} \,\square\, W(6) = \begin{cases} 6[i,j] & \text{falls } P(i,j) \\ \perp & \text{sonst} \end{cases}$

⋮

3 hoch: $\cancel{src_3^{(1)}}$ keine neuen Quellen, weil Konfliktgleichung im Widerspruch zu Ordnung

13 hoch: Ordnung: $i = i' \wedge j' < j$

⋮

6 hoch: keine Quellen, weil wie oben

⋮

3 hoch: $src^{(2)} = src^{(1)} \,\square\, W(3) = \begin{cases} 6[i,j] & \text{f. } P(i,j) \\ 3[i,j-1] \wedge \neg P(i,j) \wedge j \ge 1 \\ \perp & \text{sonst} \end{cases}$

⋮

12 hoch: Ordnung: $i' < i$; j, j' beliebig

⋮

6 hoch: Wg. Optimierung: $j = 0 \wedge \neg P(i,j)$

Konflikt: $i + 0 = i' + j' \Rightarrow j' = i - i'$

Existenz: $P(i', j') \wedge$ Indexraum

$\Rightarrow$ neue Quellen:

$6[i', i-i']$ für $j = 0 \wedge i' < i \wedge P(i', i-i')$
$\wedge \neg \exists x : P(x, i-x)$ mit $x > i'$

$$\Rightarrow src^{(3)} = \begin{cases} 6[i,j] & \text{f. } P(i,j) \\ 3[i,j-1] & \text{f. } j \geq 1 \wedge \neg P(i,j) \\ 6[i', i-i'] & \text{f. } j = 0 \wedge \neg P(x, i-x) \text{ f.a. } i' < x \leq i \\ \perp & \text{sonst} \end{cases}$$

⋮

3 hoch: Optimierung: $j = 0 \wedge \neg P(i,j) \wedge i' = i - 1$
$\Downarrow$
$j' = 0$

$\rightarrow$ neue Quellen:

$3[i-1, 0]$ für $\neg P(i,j)$, $i \geq 1$, $\neg P(i-1, 1)$

$$\Rightarrow src^{(4)} = \begin{cases} 6[i,j] & \text{f. } P(i,j) \\ 3[i,j-1] & \text{f. } j \geq 1 \wedge \neg P(i,j) \\ 6[i-1, 1] & \text{f. } j = 0 \wedge \neg P(i,j) \wedge P(i-1, 1),\ i \geq 1 \\ 3[i-1, 0] & \text{f. } j = 0 \wedge \neg P(i,j) \wedge \neg P(i-1, 1),\ i \geq 1 \\ \perp & \text{f. } i = 0 \wedge j = 0 \wedge \neg P(i,j) \end{cases}$$

Der Omega-Test

2017-07-03

Berücksichtigt Existenz- und Konfliktgleichungen

Phase ~~(a)~~ A Ganzzahliges Gleichungssystem

- falls ein Koeffizient betragsmäßig $= 1 \Rightarrow$ auflösen & einsetzen

Phase B: Ungleichungssystem (siehe unten)

Substitutionsverfahren: Kein Koeffizient einer Variablen des Gleichungssystems ist ± 1.

Definiere $\widehat{a \bmod b} = a - b \cdot \lfloor \frac{a}{b} + \frac{1}{2} \rfloor$

1. Sei k der Index der Variable mit betragsmäßig kleinstem Koeffizienten mit $a_k \neq 0$.
2. Setze $m := |a_k| + 1$
3. Führe neue Variable σ ein mit $m \cdot \sigma = \sum_i (\widehat{a_i \bmod m}) \cdot x_i$
4. Es gilt: $\widehat{a_k \bmod m} = -\operatorname{sgn}(a_k)$
5. Es gilt: $x_k = -\operatorname{sgn}(a_k) \cdot m \cdot \sigma + \sum_{i \in V \setminus \{k\}} \operatorname{sgn}(a_k) \cdot (\widehat{a_i \bmod m}) \cdot x_i$
6. Es gilt: $-|a_k| \cdot \sigma + \sum_{i \in V \setminus \{k\}} \left(\left(\lfloor \frac{a_i}{m} + \frac{1}{2} \rfloor \right) + (\widehat{a_i \bmod m}) \right) \cdot x_i = 0$

7.

Beispiel:

(1) $7x + 12y + 31z - 17 = 0$ $\qquad 1 \leq x \leq 40$

(2) $3x + 5y + 14z - 7 = 0$

$a_k = 3,\ m = 4$, y: $\widehat{5 \bmod 4} = (5 - 4 \lfloor \frac{5}{4} + \frac{1}{2} \rfloor = 1$

z: $\widehat{14 \bmod 4} = 14 - 4 \lfloor \frac{14}{4} + \frac{1}{2} \rfloor = -2$

1: $\widehat{-7 \bmod 4} = -7 - 4 \lfloor -\frac{7}{4} + \frac{1}{2} \rfloor = 1$

neue Variable σ: $x = (-1) \cdot 4 \cdot \sigma + 1y - 2z + 1$

$7x = (-28) \cdot \sigma + 7y - 14z + 7$

(2) $-3\sigma + 2y + 2z - 1 = 0$

(1) $-28\sigma + 19y + 17z - 10 = 0$ $\qquad 1 \leq -4\sigma - y - 2z + 1 \leq 40$

$a_k = 2,\ m = 3$

σ: $\widehat{-3 \bmod 3} = (-3) - (3) \cdot \lfloor -\frac{3}{3} + \frac{1}{2} \rfloor = 0$

z: $\widehat{2 \bmod 3} = 2 - 3 \lfloor \frac{2}{3} + \frac{1}{2} \rfloor = -1$

1: $\widehat{-1 \bmod 3} = -3 - 1 \lfloor -\frac{1}{3} + \frac{1}{2} \rfloor = -1$

neue Variable τ:

$$y = -3\tau + 0\sigma - 1z - 1$$
$$19y = -57\tau - 19z - 19$$

$$\left.\begin{array}{l} -2\tau - \sigma - 1 = 0 \\ -57\tau - 28\sigma - 2z - 29 = 0 \\ 1 \leq -4\sigma - 3z \overset{-3\tau}{\leq} 40 \end{array}\right\}$$

$$\sigma = -2\tau - 1$$

$$\left.\begin{array}{l} -\tau - 2z - 1 = 0 \\ 1 \leq 5\tau - 3z + 4 \leq 40 \iff -3 \leq 5\tau - 3z \leq 36 \end{array}\right\} \text{neues System}$$

$$\tau = -2z - 1$$

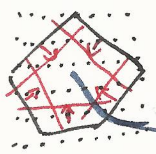
$$-3 \leq -13z - 5 \leq 36 \iff \boxed{2 \leq -13z \leq 41}$$

Also nur noch Ungleichungssystem, damit Phase B.

1. Gegengleiche Ungleichung → Gleichung → zurück in Phase A.
2. Ungleichungssystem unlösbar
   ⇒ es gibt keine Abhängigkeit ⇒ Stop.
3. Wenn nur <u>eine</u> Variable, es existiert Abhängigkeit → Stop.
4. Ansonsten FM-Varianten

<u>Problem:</u> FM arbeitet nicht ganzzahlig, d.h.

- FM sagt unlösbar ⇒ insbesondere unlösbar für ganze Zahlen
- FM sagt lösbar ⇒ 2 Fälle



wenn dunkler Schatten reell (FM) lösbar ⇒ original ganzzahlig lösbar

dunkler Schatten

sonst Nightmare, d.h. exponentiell viele Streifen einlegen und schauen ob Punkt getroffen wird.