Bäume, Ordnungen und Anwendungen

Roland Meyer

TU Kaiserslautern

Table of Contents I

1 Intraprozedurale Datenflussanalyse

Klassifikation von Datenflussanalysen

Datenflussanalysen lassen sich anhand von vier Parametern klassifizieren:

Richtung der Analyse:

Vorwärts Berechne Information über die Vergangenheit von Daten.

Rückwärts Berechne Information über das zukünftige Verhalten von Daten.

Approximation der Information:

May Überapproximiere die Information über Daten.

May-Analysen spiegeln jede Information wider, die (möglicherweise) in einem realen Ablauf eintreten kann.

Damit können May-Informationen nicht verletzt werden.

Allerdings ist nicht garantiert, dass eine Information auch in einem realen Ablauf erreicht wird.

Must Unterapproximiere die Information über Daten.

Must-Analysen spiegeln nur Information wider, die definitiv in jedem realen Ablauf eintritt.

Damit liefern Must-Analysen verlässlich eintretende Informationen.

Allerdings geben Must-Analysen nicht alle eintretenden Informationen wieder.

Klassifikation von Datenflussanalysen

Berücksichtigung von **Prozeduren**:

Intraprozedural Analyse einer einzelnen Prozedur, typischerweise `main`.

Um Programme intraprozedural zu analysieren, nutze Inlining.

Inlining ist bei Rekursion nicht möglich. Intraprozedurale Analysen unterstützen keine Rekursion.

Interprozedural Analyse eines ganzen Programms mit Rekursion.

Berücksichtigung des **Kontrollflusses**:

Control-flow sensitive Berücksichtige die Anordnung der Befehle im Programm.

Die Analyse berechnet separate Information für jeden Block.

Vorteil: präzise. Nachteil: ineffizient.

Control-flow insensitive Vergiss die Anordnung der Befehle im Programm.

Die Analyse berechnet eine Information für alle Blöcke.

Vorteil: effizient. Nachteil: unpräzise.

Klassifikation von Datenflussanalysen

Wir betrachten vier klassische Analysen, die alle vier Kombinationen aus Richtung und Approximation abdecken.

Allerdings sind alle vier Analysen conrol-flow sensitiv und intraprozedural.

Folgende Tabelle zeigt die Analysen und den Zusammenhang zwischen:

$$\begin{aligned} \text{Richtung} \quad &\leftrightarrow \quad \text{Wahl des Kontrollflussgraphen mit Extremalknoten} \\ \text{Approximation} \quad &\leftrightarrow \quad \text{Wahl des Verbandes mit Join und Bottom.} \end{aligned}$$

Instanz	Reaching-Definitions	Available-Expr.	Live-Var.	Busy-Expr.
Richtung	vorwärts		rückwärts	
Extremal (E)	initaler Block		finale Blöcke	
Fluss. (F)	in Programmordnung		gegen Programmordnung	
Approx.	may	must	may	must
Verband	$(\mathbb{P}(Vars \times Blocks \cup \{?\}), \subseteq)$	$(\mathbb{P}(AExp), \supseteq)$	$(\mathbb{P}(Vars), \subseteq)$	$(\mathbb{P}(AExp), \supseteq)$
Join ($\sqcup$)	$\cup$	$\cap$	$\cup$	$\cap$
Bottom ($\bot$)	$\emptyset$	$AExp$	$\emptyset$	$AExp$
Anfangsw. (i)	$\{(x, ?) \mid x \in Vars\}$	$\emptyset$	$Vars$	$\emptyset$
Transferf. (f)	$f_b(X) := (X \setminus kill(b)) \cup gen(b)$			

Reaching-Definitions-Analyse

Ziel:
Berechne für jeden Block die Zuweisungen, die es gegegeben haben könnte (nicht überschrieben), wenn eine Ausführung den Block erreicht.

Klassifikation:
Vorwärtsanalyse, die Information über die Vergangenheit von Daten berechnet.

May-Analyse, die das Verhalten aller einzelnen Ausführungen überapproximiert.
Das heißt, das Verhalten jeder Ausführung ist sicher in der Information enthalten.

Idee: Tafel.

Anwendungen:
Berechnung von Use-Definition-Chains, die angeben, welche Zuweisungen (Definitions) von einem Block genutzt werden.

Use-Definition-Chains sind die Grundlage für Code-Motion-Optimierungen.

Reaching-Definitions-Analyse

Betrachte ein Programm mit Variablen *Vars* und Blöcken *Blocks*.

Definiere das Datenflussystem $S = (G, (D, \preceq), i, \{f_b : D \to D \mid b \in Blocks\})$.

Kontrollflussgraph $G = (B, E, F)$:
$B = Blocks$, $E =$ initialer Block, $F =$ Kontrollfluss in Programmordnung.

Verband $(D, \preceq)$:
$(D, \preceq) = (\mathbb{P}(Vars \times (Blocks \cup \{?\})), \subseteq)$.

Es handelt sich um einen (Potenzmengen)verband.

(ACC) gilt, da der Verband beschränkte Höhe hat.

Die Bedeutung der Elemente in $Vars \times (Blocks \cup \{?\})$ ist wie folgt:

$$\begin{aligned} (x, ?) &= x \text{ ist möglicherweise noch nicht initialisiert.} \\ (x, b) &= x \text{ hat möglicherweise die letzte Zuweisung von Block } b \text{ erhalten.} \end{aligned}$$

Anfangswert i:
$\{(x, ?) \mid x \in Vars\}$.

Reaching-Definitions-Analyse

Transferfunktionen $f_b : D \to D$:

$$f_b : \mathbb{P}(Vars \times (Blocks \cup \{?\})) \to \mathbb{P}(Vars \times (Blocks \cup \{?\}))$$
$$X \mapsto (X \setminus kill(b)) \cup gen(b)$$

Die Mengen $kill(b), gen(b) \subseteq Vars \times (Blocks \cup \{?\})$ sind

$$kill(b) := \begin{cases} \{(x, ?)\} \cup \{(x, b) \mid b \in Blocks\}, & \text{falls } b = [x := a]^l \\ \emptyset, & \text{sonst.} \end{cases}$$

//Zuweisungen, die von Block b überschrieben werden.

$$gen(b) := \begin{cases} \{(x, b)\}, & \text{falls } b = [x := a]^l \\ \emptyset, & \text{sonst.} \end{cases}$$

//Zuweisungen, die von Block b generiert werden.

Die Transferfunktionen sind monoton.

Reaching-Definitions-Analyse

Betrachte das Beispielprogramm c an der Tafel. Die Transferfunktionen sind

Block	$kill(b)$	$gen(b)$	$f_b(X)$
$[x := 5]^1$	$\{(x,?),(x,1),(x,5)\}$	$\{(x,1)\}$	$(X \setminus \{(x,?),(x,1),(x,5)\}) \cup \{(x,1)\}$
$[y := 1]^2$	$\{(y,?),(y,2),(y,4)\}$	$\{(y,2)\}$	$(X \setminus \{(y,?),(y,2),(y,4)\}) \cup \{(y,2)\}$
$[x > 1]^3$	$\emptyset$	$\emptyset$	X
$[y := xy]^4$	$\{(y,?),(y,2),(y,4)\}$	$\{(y,4)\}$	$(X \setminus \{(y,?),(y,2),(y,4)\}) \cup \{(y,4)\}$
$[x := x-1]^5$	$\{(x,?),(x,1),(x,5)\}$	$\{(x,5)\}$	$(X \setminus \{(x,?),(x,1),(x,5)\}) \cup \{(x,5)\}$

In der Tabelle sind die $gen(b)$ Mengen auf die Blöcke eingeschränkt worden, die eine Zuweisung auf die Variable durchführen.

Das vom Datenflusssystem induzierte Gleichungssystem ist

$$X_1 = \underbrace{\{(x,?),(y,?)\}}_{=i}$$

$$X_2 = \underbrace{(X_1 \setminus \{(x,?),(x,1),(x,5)\}) \cup \{(x,1)\}}_{=f_1(X_1)}$$

$$X_3 = \underbrace{((X_2 \setminus \{(y,?),(y,2),(y,4)\}) \cup \{(y,2)\})}_{=f_2(X_2)} \cup \underbrace{((X_5 \setminus \{(x,?),(x,1),(x,5)\}) \cup \{(x,5)\})}_{=f_5(X_5)}$$

$$X_4 = X_3$$

$$X_5 = (X_4 \setminus \{(y,?),(y,2),(y,4)\}) \cup \{(y,4)\}$$

Reaching-Definitions-Analyse

$$
\begin{aligned}
X_1 &= \{(x,?),(y,?)\}\\
X_2 &= (X_1 \setminus \{(x,?),(x,1),(x,5)\}) \cup \{(x,1)\}\\
X_3 &= ((X_2 \setminus \{(y,?),(y,2),(y,4)\}) \cup \{(y,2)\}) \cup ((X_5 \setminus \{(x,?),(x,1),(x,5)\}) \cup \{(x,5)\})\\
X_4 &= X_3\\
X_5 &= (X_4 \setminus \{(y,?),(y,2),(y,4)\}) \cup \{(y,4)\}
\end{aligned}
$$

Berechne eine Lösung des Gleichungssystems durch Iteration von

$$g_S : \mathbb{P}(\mathit{Vars} \times (\mathit{Blocks} \cup \{?\}))^5 \to \mathbb{P}(\mathit{Vars} \times (\mathit{Blocks} \cup \{?\}))^5$$

auf $\perp$ von $(\mathbb{P}(\mathit{Vars} \times (\mathit{Blocks} \cup \{?\}))^5, \subseteq^5)$ bis zum kleinsten Fixpunkt:

Iter.	$(d_1$	d_2	d_3	d_4	$d_5)$
$g_S^0(\perp)$	$(\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset)$
$g_S^1(\perp)$	$(\{(x,?),(y,?)\}$	$\{(x,1)\}$	$\{(y,2),(x,5)\}$	$\emptyset$	$\{(y,4)\})$
$g_S^2(\perp)$	$(\{(x,?),(y,?)\}$	$\{(y,?),(x,1)\}$	$\{(x,1),(y,2),(y,4),(x,5)\}$	$\{(y,2),(x,5)\}$	$\{(y,4)\})$
$g_S^3(\perp)$	$(\{(x,?),(y,?)\}$	$\{(y,?),(x,1)\}$	$\{(x,1),(y,2),(y,4),(x,5)\}$	$\{(x,1),(y,2),(y,4),(x,5)\}$	$\{(x,5)(y,4)\})$
$g_S^4(\perp)$	$(\{(x,?),(y,?)\}$	$\{(y,?),(x,1)\}$	$\{(x,1),(y,2),(y,4),(x,5)\}$	$\{(x,1),(y,2),(y,4),(x,5)\}$	$\{(x,1),(x,5),(y,4)\}$
$g_S^5(\perp)$	$(\{(x,?),(y,?)\}$	$\{(y,?),(x,1)\}$	$\{(x,1),(y,2),(y,4),(x,5)\}$	$\{(x,1),(y,2),(y,4),(x,5)\}$	$\{(x,1),(x,5),(y,4)\}$

Es gilt $g_S(g_S^4(\perp)) = g_S^4(\perp)$. Also ist $g_S^4(\perp)$ der kleinste Fixpunkt.

Reaching-Definitions-Analyse

Iter.	$(d_1$	d_2	d_3	d_4	$d_5)$
$g_S^0(\bot)$	$(\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset)$
$g_S^1(\bot)$	$(\{(x,?),(y,?)\}$	$\{(x,1)\}$	$\{(y,2),(x,5)\}$	$\emptyset$	$\{(y,4)\})$
$g_S^2(\bot)$	$(\{(x,?),(y,?)\}$	$\{(y,?),(x,1)\}$	$\{(x,1),(y,2),(y,4),(x,5)\}$	$\{(y,2),(x,5)\}$	$\{(y,4)\})$
$g_S^3(\bot)$	$(\{(x,?),(y,?)\}$	$\{(y,?),(x,1)\}$	$\{(x,1),(y,2),(y,4),(x,5)\}$	$\{(x,1),(y,2),(y,4),(x,5)\}$	$\{(x,5)(y,4)\})$
$g_S^4(\bot)$	$(\{(x,?),(y,?)\}$	$\{(y,?),(x,1)\}$	$\{(x,1),(y,2),(y,4),(x,5)\}$	$\{(x,1),(y,2),(y,4),(x,5)\}$	$\{(x,1),(x,5),(y,4)\}$
$g_S^5(\bot)$	$(\{(x,?),(y,?)\}$	$\{(y,?),(x,1)\}$	$\{(x,1),(y,2),(y,4),(x,5)\}$	$\{(x,1),(y,2),(y,4),(x,5)\}$	$\{(x,1),(x,5),(y,4)\}$

Die kleinste Lösung des Gleichungssystems ist

$$X_1 = \{(x,?),(y,?)\} \qquad X_2 = \{(y,?),(x,1)\}$$
$$X_3 = \{(x,1),(y,2),(y,4),(x,5)\} \qquad X_4 = \{(x,1),(y,2),(y,4),(x,5)\}$$
$$X_5 = \{(x,1),(x,5),(y,4)\}.$$

Die kleinste Lösung ist die gewünschte Information.

Größere May-Information bedeutet Informationsverlust.

Available-Expressions-Analyse

Ziel:
Berechne für jeden Block die Ausdrücke, die auf allen Pfaden zu dem Block definitiv berechnet worden sind (nicht zwischendurch geändert).

Klassifikation:
Vorwärtsanalyse, die Information über die Vergangenheit von Daten berechnet.

Must-Analyse, die das gemeinsame Verhalten aller Ausführungen unterapproximiert.

Das heißt, die berechnete Information gilt definitiv für alle Ausführungen.

Idee: Tafel.

Anwendungen:
Vermeide erneute Berechnung bekannter Werte.

Available-Expressions-Analyse

Betrachte ein Programm mit Teilausdrücken *AExp* und Blöcken *Blocks*.
Nutze $AExp(a)$ für die Teilausdrücke von $a \in AExp$.
Nutze $Vars(a)$ für die Variablen von $a \in AExp$.

Definiere das Datenflussystem $S = (G, (D, \preceq), i, \{f_b : D \to D \mid b \in Blocks\})$.

Kontrollflussgraph $G = (B, E, F)$:
$B = Blocks$, $E =$ initialer Block, $F =$ Kontrollfluss in Programmordnung.

Verband $(D, \preceq)$:
$(D, \preceq) = (\mathbb{P}(AExp), \supseteq)$.

Es handelt sich um einen (dualen Potenzmengen)verband.

(ACC) gilt, da der Verband beschränkte Höhe hat.

Anfangswert i:
$\emptyset$.

Available-Expressions-Analyse

Transferfunktionen $f_b : D \to D$:

$$f_b : \mathbb{P}(AExp) \to \mathbb{P}(AExp)$$
$$X \mapsto (X \setminus kill(b)) \cup gen(b)$$

Die Mengen $kill(b), gen(b) \subseteq AExp$ sind

$$kill(b) := \begin{cases} \{a' \in AExp \mid x \in Vars(a')\}, & \text{falls } b = [x := a]^l \\ \emptyset, & \text{sonst.} \end{cases}$$

//Teilausdrücke, die x enthalten und daher von Block b geändert werden.

$$gen(b) := \begin{cases} \{a' \in AExp(a) \mid x \notin Vars(a')\}, & \text{falls } b = [x := a]^l \\ AExp(cond), & \text{falls } b = [cond]^l \\ \emptyset, & \text{sonst.} \end{cases}$$

//Teilausdrücke, die von Block b genutzt werden.

Die Transferfunktionen sind monoton.

Available-Expressions-Analyse

Betrachte das Beispielprogramm c an der Tafel. Die Transferfunktionen sind

Block	$kill(b)$	$gen(b)$	$f_b(X)$
$[x := a+b]^1$	$\emptyset$	$\{a+b\}$	$X \cup \{a+b\}$
$[y := ab]^2$	$\emptyset$	$\{ab\}$	$X \cup \{ab\}$
$[y > a+b]^3$	$\emptyset$	$\{a+b\}$	$X \cup \{a+b\}$
$[a := a+1]^4$	$\{a+b, ab, a+1\}$	$\emptyset$	$X \setminus \{a+b, ab, a+1\}$
$[x := a+b]^5$	$\emptyset$	$\{a+b\}$	$X \cup \{a+b\}$

Das vom Datenflusssystem induzierte Gleichungssystem ist

$$
\begin{aligned}
X_1 &= \underbrace{\emptyset}_{=i} \\
X_2 &= \underbrace{X_1 \cup \{a+b\}}_{=f_1(X_1)} \\
X_3 &= \underbrace{(X_2 \cup \{ab\})}_{=f_2(X_2)} \cap \underbrace{(X_5 \cup \{a+b\})}_{=f_5(X_5)} \\
X_4 &= X_3 \cup \{a+b\} \\
X_5 &= X_4 \setminus \{a+b, ab, a+1\}
\end{aligned}
$$

Available-Expressions-Analyse

$$\begin{aligned}
X_1 &= \emptyset \\
X_2 &= X_1 \cup \{a+b\} \\
X_3 &= (X_2 \cup \{ab\}) \cap (X_5 \cup \{a+b\}) \\
X_4 &= X_3 \cup \{a+b\} \\
X_5 &= X_4 \setminus \{a+b, ab, a+1\}
\end{aligned}$$

Berechne eine Lösung des Gleichungssystems durch Iteration von

$$g_S : \mathbb{P}(AExp)^5 \to \mathbb{P}(AExp)^5$$

auf $\perp$ von $(\mathbb{P}(AExp)^5, \supseteq^5)$ bis zum kleinsten Fixpunkt:

Iter.	$(d_1$	d_2	d_3	d_4	$d_5)$
$g_S^0(\perp)$	$(\{a+b, ab, a+1\}$	$\{a+b, ab, a+1\}$	$\{a+b, ab, a+1\}$	$\{a+b, ab, a+1\}$	$\{a+b, ab, a+1\})$
$g_S^1(\perp)$	$(\emptyset$	$\{a+b, ab, a+1\}$	$\{a+b, ab, a+1\}$	$\{a+b, ab, a+1\}$	$\emptyset)$
$g_S^2(\perp)$	$(\emptyset$	$\{a+b\}$	$\{a+b\}$	$\{a+b, ab, a+1\}$	$\emptyset)$
$g_S^3(\perp)$	$(\emptyset$	$\{a+b\}$	$\{a+b\}$	$\{a+b\}$	$\emptyset)$
$g_S^4(\perp)$	$(\emptyset$	$\{a+b\}$	$\{a+b\}$	$\{a+b\}$	$\emptyset)$

Es gilt $g_S(g_S^3(\perp)) = g_S^3(\perp)$. Also ist $g_S^3(\perp)$ der kleinste Fixpunkt.

Available-Expressions-Analyse

Iter.	$(d_1$	d_2	d_3	d_4	$d_5)$
$g_S^0(\bot)$	$(\{a+b, ab, a+1\}$	$\{a+b, ab, a+1\}$	$\{a+b, ab, a+1\}$	$\{a+b, ab, a+1\}$	$\{a+b, ab, a+1\})$
$g_S^1(\bot)$	$(\emptyset$	$\{a+b, ab, a+1\}$	$\{a+b, ab, a+1\}$	$\{a+b, ab, a+1\}$	$\emptyset)$
$g_S^2(\bot)$	$(\emptyset$	$\{a+b\}$	$\{a+b\}$	$\{a+b, ab, a+1\}$	$\emptyset)$
$g_S^3(\bot)$	$(\emptyset$	$\{a+b\}$	$\{a+b\}$	$\{a+b\}$	$\emptyset)$
$g_S^4(\bot)$	$(\emptyset$	$\{a+b\}$	$\{a+b\}$	$\{a+b\}$	$\emptyset)$

Die kleinste Lösung des Gleichungssystems ist

$$X_1 = \emptyset = X_5 \qquad\qquad X_2 = \{a+b\} = X_3 = X_4.$$

Die kleinste Lösung ist die gewünschte Information.

Größere (bzgl. $\supseteq$) Must-Information bedeutet Informationsverlust.

Bemerkung: Wir haben hier den größten Fixpunkt auf dem Potenzmengenverband $(\mathbb{P}(AExp), \subseteq)$ berechnet.

Durch Dualisierung des Verbandes zu $(\mathbb{P}(AExp), \supseteq)$ konnten wir eine kleinste Fixpunktberechnung und so unser Framework mit (ACC) nutzen.

Live-Variables-Analyse

Definition:
Eine Variable heißt lebendig am Ausgang eines Blocks, falls es einen Ablauf von diesem Block zu einem anderen Block geben könnte (nicht überschrieben), der die Variable in einer Bedingung oder Zuweisung (rechte Seite) nutzt.

Am Ende des Programms sind alle Variablen lebendig.

Ziel:
Berechne für jeden Block die Variablen, die am Ausgang lebendig sind.

Klassifikation:
Rückwärtsanalyse, die Information über die Zukunft von Daten berechnet.

May-Analyse, die das Verhalten aller einzelnen Ausführungen überapproximiert. Das heißt, das Verhalten jeder Ausführung ist sicher in der Information enthalten.

Idee: Tafel.

Live-Variables-Analyse

Anwendungen:

Register-Allocation: Falls x lebendig ist, wird die Variable vermutlich bald genutzt und sollte ein Register erhalten.

Ist x nicht mehr lebendig, kann das Register neu vergeben werden.

Dead-Code-Elimination: Ist x am Ausgang einer Zuweisung (zu x) nicht lebendig, kann die Zuweisung entfernt werden.

Auf ähnliche Weise lassen sich Variablen zusammenfassen: sind x und y nie gemeinsam lebendig, verwende eine Variable z.

Live-Variables-Analyse

Betrachte ein Programm mit Variablen Blöcken *Blocks* und Variablen *Vars*. Ferner sei *Vars*(*a*) die Menge der Variablen in einem Ausdruck *a*.

Definiere das Datenflussystem $S = (G, (D, \preceq), i, \{f_b : D \to D \mid b \in Blocks\})$.

Kontrollflussgraph $G = (B, E, F)$:
$B = Blocks$, $E =$ finale Blöcke, $F =$ Kontrollfluss gegen die Programmordnung.

Verband $(D, \preceq)$:
$(D, \preceq) = (\mathbb{P}(Vars), \subseteq)$.

Es handelt sich um einen (Potenzmengen)verband.

(ACC) gilt, da der Verband beschränkte Höhe hat.

Anfangswert i:
Vars (am Ende des Programms sind per Definition alle Variablen lebendig).

Live-Variables-Analyse

Transferfunktionen $f_b : D \to D$:

$$f_b : \mathbb{P}(Vars) \to \mathbb{P}(Vars)$$
$$X \mapsto (X \setminus kill(b)) \cup gen(b)$$

Die Mengen $kill(b), gen(b) \subseteq Vars$ sind

$$kill(b) := \begin{cases} \{x\}, & \text{falls } b = [x := a]^l \\ \emptyset, & \text{sonst.} \end{cases}$$

//Variablen, die von Block b überschrieben werden.

$$gen(b) := \begin{cases} Vars(a), & \text{falls } b = [x := a]^l \\ Vars(cond), & \text{falls } b = [cond]^l \\ \emptyset, & \text{sonst.} \end{cases}$$

//Variablen, die von Block b genutzt werden.

Die Transferfunktionen sind monoton.

Live-Variables-Analyse

Betrachte das Beispielprogramm c an der Tafel. Die Transferfunktionen sind

Block	$kill(b)$	$gen(b)$	$f_b(X)$
$[x := 2]^1$	$\{x\}$	$\emptyset$	$X \setminus \{x\}$
$[y := 4]^2$	$\{y\}$	$\emptyset$	$X \setminus \{y\}$
$[x := 1]^3$	$\{x\}$	$\emptyset$	$X \setminus \{x\}$
$[y > 0]^4$	$\emptyset$	$\{y\}$	$X \cup \{y\}$
$[z := x]^5$	$\{z\}$	$\{x\}$	$(X \setminus \{z\}) \cup \{x\}$
$[z := yy]^6$	$\{z\}$	$\{y\}$	$(X \setminus \{z\}) \cup \{y\}$
$[x := z]^7$	$\{x\}$	$\{z\}$	$(X \setminus \{x\}) \cup \{z\}$

Das vom Datenflusssystem induzierte Gleichungssystem ist

$$\begin{aligned}
X_1 &= X_2 \setminus \{y\} \\
X_2 &= X_3 \setminus \{x\} \\
X_3 &= X_4 \cup \{y\} \\
X_4 &= \underbrace{((X_5 \setminus \{z\}) \cup \{x\})}_{=f_5(X_5)} \cup \underbrace{((X_6 \setminus \{z\}) \cup \{y\})}_{=f_6(X_6)} \\
X_5 &= (X_7 \setminus \{x\}) \cup \{z\} \\
X_6 &= (X_7 \setminus \{x\}) \cup \{z\} \\
X_7 &= \underbrace{\{x, y, z\}}_{=i}
\end{aligned}$$

Live-Variables-Analyse

$$
\begin{aligned}
X_1 &= X_2 \setminus \{y\} \\
X_2 &= X_3 \setminus \{x\} \\
X_3 &= X_4 \cup \{y\} \\
X_4 &= ((X_5 \setminus \{z\}) \cup \{x\}) \cup ((X_6 \setminus \{z\}) \cup \{y\}) \\
X_5 &= (X_7 \setminus \{x\}) \cup \{z\} \\
X_6 &= (X_7 \setminus \{x\}) \cup \{z\} \\
X_7 &= \{x, y, z\}
\end{aligned}
$$

Berechne eine Lösung des Gleichungssystems durch Iteration von

$$g_S : \mathbb{P}(\mathit{Vars})^7 \to \mathbb{P}(\mathit{Vars})^7$$

auf $\perp$ von $(\mathbb{P}(\mathit{Vars})^7, \subseteq^7)$ bis zum kleinsten Fixpunkt:

Iter.	$(d_1$	d_2	d_3	d_4	d_5	d_6	$d_7)$
$g_S^0(\perp)$	$(\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset)$
$g_S^1(\perp)$	$(\emptyset$	$\emptyset$	$\{y\}$	$\{y, x\}$	$\{z\}$	$\{z\}$	$\{x, y, z\})$
$g_S^2(\perp)$	$(\emptyset$	$\{y\}$	$\{y, x\}$	$\{y, x\}$	$\{y, z\}$	$\{y, z\}$	$\{x, y, z\})$
$g_S^3(\perp)$	$(\emptyset$	$\{y\}$	$\{y, x\}$	$\{y, x\}$	$\{y, z\}$	$\{y, z\}$	$\{x, y, z\})$

Es gilt $g_S(g_S^2(\perp)) = g_S^2(\perp)$. Also ist $g_S^2(\perp)$ der kleinste Fixpunkt.

Live-Variables-Analyse

Iter.	$(d_1$	d_2	d_3	d_4	d_5	d_6	$d_7)$
$g_S^0(\bot)$	$(\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset)$
$g_S^1(\bot)$	$(\emptyset$	$\emptyset$	$\{y\}$	$\{y,x\}$	$\{z\}$	$\{z\}$	$\{x,y,z\})$
$g_S^2(\bot)$	$(\emptyset$	$\{y\}$	$\{y,x\}$	$\{y,x\}$	$\{y,z\}$	$\{y,z\}$	$\{x,y,z\})$
$g_S^3(\bot)$	$(\emptyset$	$\{y\}$	$\{y,x\}$	$\{y,x\}$	$\{y,z\}$	$\{y,z\}$	$\{x,y,z\})$

Die kleinste Lösung des Gleichungssystems ist

$$
\begin{aligned}
X_1 &= \emptyset & X_2 &= \{y\} \\
X_3 &= \{y,x\} = X_4 & X_5 &= \{y,z\} = X_6 \\
X_7 &= \{x,y,z\}.
\end{aligned}
$$

Die kleinste Lösung ist die gewünschte Information.

Größere May-Information bedeutet Informationsverlust.

Der Block $[x := 2]^1$ kann entfernt werden.

Very-Busy-Expressions-Analyse

Definition:
Ein Ausdruck heißt very busy am Ausgang eines Blocks, falls der Ausdruck auf jedem Pfad, der von diesem Block ausgeht, verwendet wird, bevor eine der enthaltenen Variablen neu geschrieben wird.

Ziel:
Berechne für jeden Block die Ausdrücke, die am Ausgang very busy sind.

Klassifikation:
Rückwärtsanalyse, die Information über die Zukunft von Daten berechnet.

Must-Analyse, die das gemeinsame Verhalten aller Ausführungen unterapproximiert.

Das heißt, die berechnete Information gilt definitiv für alle Ausführungen.

Idee: Tafel.

Anwendungen:
Hoisting-Expressions: Betrachte eine Schleife mit einem Block $x := (a + b)y$, wobei $a + b$ von der Schleife nicht geändert wird. Dann lässt sich eine Zuweisung $t := a + b$ vor der Schleife einfügen und $x := (a + b)y$ durch $x := ty$ ersetzen.

Very-Busy-Expressions-Analyse

Betrachte ein Programm mit Teilausdrücken $AExp$ und Blöcken $Blocks$.
Nutze $AExp(a)$ für die Teilausdrücke von $a \in AExp$.
Nutze $Vars(a)$ für die Variablen von $a \in AExp$.

Definiere das Datenflussystem $S = (G, (D, \preceq), i, \{f_b : D \to D \mid b \in Blocks\})$.

Kontrollflussgraph $G = (B, E, F)$:
$B = Blocks$, $E =$ finale Blöcke, $F =$ Kontrollfluss gegen die Programmordnung.

Verband $(D, \preceq)$:
$(D, \preceq) = (\mathbb{P}(AExp), \supseteq)$.

Es handelt sich um einen (dualen Potenzmengen)verband.

(ACC) gilt, da der Verband beschränkte Höhe hat.

Anfangswert i:
$\emptyset$.

Very-Busy-Expressions-Analyse

Transferfunktionen $f_b : D \to D$:

$$f_b : \mathbb{P}(AExp) \to \mathbb{P}(AExp)$$
$$X \mapsto (X \setminus \mathit{kill}(b)) \cup \mathit{gen}(b)$$

Die Mengen $\mathit{kill}(b), \mathit{gen}(b) \subseteq AExp$ sind

$$\mathit{kill}(b) := \begin{cases} \{a' \in AExp \mid x \in Vars(a')\}, & \text{falls } b = [x := a]^l \\ \emptyset, & \text{sonst.} \end{cases}$$

//Teilausdrücke, die x enthalten und daher von Block b geändert werden.

$$\mathit{gen}(b) := \begin{cases} AExp(a), & \text{falls } b = [x := a]^l \\ AExp(cond), & \text{falls } b = [cond]^l \\ \emptyset, & \text{sonst.} \end{cases}$$

//Teilausdrücke, die von Block b genutzt werden.

Die Transferfunktionen sind monoton.

Very-Busy-Expressions-Analyse

Betrachte das Beispielprogramm c an der Tafel. Die Transferfunktionen sind

Block	$kill(b)$	$gen(b)$	$f_b(X)$
$[a > b]^1$	$\emptyset$	$\emptyset$	X
$[x := b - a]^2$	$\emptyset$	$\{b - a\}$	$X \cup \{b - a\}$
$[y := a - b]^3$	$\emptyset$	$\{a - b\}$	$X \cup \{a - b\}$
$[y := b - a]^4$	$\emptyset$	$\{b - a\}$	$X \cup \{b - a\}$
$[x := a - b]^5$	$\emptyset$	$\{a - b\}$	$X \cup \{a - b\}$

Das vom Datenflusssystem induzierte Gleichungssystem ist

$$X_1 = \underbrace{(X_2 \cup \{b - a\})}_{=f_2(X_2)} \cap \underbrace{(X_4 \cup \{b - a\})}_{=f_4(X_4)}$$

$$X_2 = X_3 \cup \{a - b\}$$

$$X_3 = \underbrace{\emptyset}_{=i}$$

$$X_4 = X_5 \cup \{a - b\}$$

$$X_5 = \underbrace{\emptyset}_{=i}$$

Very-Busy-Expressions-Analyse

$$
\begin{aligned}
X_1 &= (X_2 \cup \{b - a\}) \cap (X_4 \cup \{b - a\}) \\
X_2 &= X_3 \cup \{a - b\} \\
X_3 &= \emptyset \\
X_4 &= X_5 \cup \{a - b\} \\
X_5 &= \emptyset
\end{aligned}
$$

Berechne eine Lösung des Gleichungssystems durch Iteration von

$$g_S : \mathbb{P}(AExp)^5 \to \mathbb{P}(AExp)^5$$

auf $\perp$ von $(\mathbb{P}(AExp)^5, \supseteq^5)$ bis zum kleinsten Fixpunkt:

Iter.	$(d_1$	d_2	d_3	d_4	$d_5)$
$g_S^0(\perp)$	$(\{a-b, b-a\}$	$\{a-b, b-a\}$	$\{a-b, b-a\}$	$\{a-b, b-a\}$	$\{a-b, b-a\})$
$g_S^1(\perp)$	$(\{a-b, b-a\}$	$\{a-b, b-a\}$	$\emptyset$	$\{a-b, b-a\}$	$\emptyset)$
$g_S^2(\perp)$	$(\{a-b, b-a\}$	$\{a-b\}$	$\emptyset$	$\{a-b\}$	$\emptyset)$
$g_S^3(\perp)$	$(\{a-b, b-a\}$	$\{a-b\}$	$\emptyset$	$\{a-b\}$	$\emptyset)$

Es gilt $g_S(g_S^2(\perp)) = g_S^2(\perp)$. Also ist $g_S^2(\perp)$ der kleinste Fixpunkt.

Very-Busy-Expressions-Analyse

Iter.	$(d_1$	d_2	d_3	d_4	$d_5)$
$g_S^0(\bot)$	$(\{a-b, b-a\}$	$\{a-b, b-a\}$	$\{a-b, b-a\}$	$\{a-b, b-a\}$	$\{a-b, b-a\})$
$g_S^1(\bot)$	$(\{a-b, b-a\}$	$\{a-b, b-a\}$	$\emptyset$	$\{a-b, b-a\}$	$\emptyset)$
$g_S^2(\bot)$	$(\{a-b, b-a\}$	$\{a-b\}$	$\emptyset$	$\{a-b\}$	$\emptyset)$
$g_S^3(\bot)$	$(\{a-b, b-a\}$	$\{a-b\}$	$\emptyset$	$\{a-b\}$	$\emptyset)$

Die kleinste Lösung des Gleichungssystems ist

$$X_1 = \{a-b, b-a\} \qquad X_2 = \{a-b\} = X_4 \qquad X_3 = \emptyset = X_5.$$

Die kleinste Lösung ist die gewünschte Information.

Größere (bzgl. $\supseteq$) Must-Information bedeutet Informationsverlust.

Bemerkung: Wir haben hier den größten Fixpunkt auf dem Potenzmengenverband $(\mathbb{P}(AExp), \subseteq)$ berechnet.

Durch Dualisierung des Verbandes zu $(\mathbb{P}(AExp), \supseteq)$ konnten wir eine kleinste Fixpunktberechnung und so unser Framework mit (ACC) nutzen.

Distributive Frameworks

Werden Datenflusssysteme $S = (G, (D, \preceq), i, \{f_b : D \to D \mid b \in Blocks\})$ betrachtet, deren Transferfunktionen f_b nicht nur monoton sondern distributiv sind, dann spricht man von einem **distributiven Framework**.

In den obigen vier Beispielen nutzten alle Verbände die Domäne $(\mathbb{P}(A), \sqsubseteq)$ über einer endlichen Menge A und mit $\sqsubseteq \in \{\subseteq, \supseteq\}$.

Ferner waren die Transferfunktionen $f_b : \mathbb{P}(A) \to \mathbb{P}(A)$ definiert durch

$$f_b(X) := (X \setminus kill(b)) \cup gen(b) \qquad \text{mit} \qquad kill(b), gen(b) \subseteq A.$$

Werden nur Datenflusssysteme der Form $S = (G, (\mathbb{P}(A), \sqsubseteq), i, f)$ mit f bestehend aus Gen/Kill-Transferfunktionen betrachtet, spricht man von einem **Bitvektor-Framework**.

Der Grund für den Namen ist, dass sich die Datenflussmengen in $\mathbb{P}(A)$ als Bitvektoren darstellen lassen.

Theorem

Bitvektor-Frameworks sind distributive Frameworks.

Effizientere Fixpunktberechnung

Beobachtung:
Die Fixpunktberechnung bestimmt den Wert von X_b in jedem Schritt neu — auch wenn sich die Belegung der Variablen der Vorgängerblöcke nicht geändert hat.

Idee:
Modifiziere die Fixpunktberechnung, so dass Variablen X_b nur bei Änderung der Eingabe neu berechnet werden.

Ansatz:
Führe Worklist in die Fixpunktberechnung ein.

Effizientere Fixpunktberechnung

procedure Worklist-Algorithmus für lfp

Eingabe: Datenflusssystem $S = (G, (D, \preceq), i, f)$ mit $G = (B, E, F)$
Variablen: X_b für Blöcke $b \in B$, initial $X_b = \bot$
W Worklist, initial $W = \varepsilon$

for all $(b, b') \in F$ **do** $W := W.(b, b')$ **endfor**
for all $b \in E$ **do** $X_b := i$ **endfor**
while $W \neq \varepsilon$ **do**
  pop (b, b') **from** W;
  if $f_b(X_b) \not\preceq X_{b'}$ **then**
    $X_{b'} := X_{b'} \sqcup f_b(X_b)$;
    for all $(b', b'') \in F$ **do**
      if $(b', b'') \notin W$ **then** $W := W.(b', b'')$ **endif**
    endfor
  endif
endwhile

Ausgabe: X_b für jeden Block $b \in B$.

Effizientere Fixpunktberechnung

Theorem

Sei das Datenflusssystem S die Eingabe für obigen Algorithmus. Der Algorithmus terminiert und berechnet $lfp(g_S)$.

Effizientere Fixpunktberechnung

Available-Expressions-Analyse am Beispielprogramm mittels Worklist:

Nach Initialisierung:

$$W = (1,2).(2,3).(3,4).(4,5).(5,3)$$

$$\begin{aligned} X_1 &= \emptyset \\ X_2 &= AExp \\ X_3 &= AExp \\ X_4 &= AExp \\ X_5 &= AExp \end{aligned}$$

Es gilt $f_1(X_1) = \{a+b\} \not\supseteq AExp = X_2$, also $X_2 := AExp \cap \{a+b\}$.
Die Kante $(2,3)$ ist noch in der Worklist enthalten.

Effizientere Fixpunktberechnung

Available-Expressions-Analyse am Beispielprogramm mittels Worklist:

Nach Iteration 1:

$$W = (2,3).(3,4).(4,5).(5,3)$$

$$\begin{aligned} X_1 &= \emptyset \\ X_2 &= \{a+b\} \\ X_3 &= AExp \\ X_4 &= AExp \\ X_5 &= AExp \end{aligned}$$

Es gilt $f_2(X_2) = \{a+b, ab\} \not\supseteq AExp = X_3$, also $X_3 := AExp \cap \{a+b, ab\}$.
Die Kante $(3,4)$ ist noch in der Worklist enthalten.

Effizientere Fixpunktberechnung

Available-Expressions-Analyse am Beispielprogramm mittels Worklist:

Nach Iteration 2:

$$W = (3,4).(4,5).(5,3)$$

$$\begin{aligned} X_1 &= \emptyset \\ X_2 &= \{a+b\} \\ X_3 &= \{a+b, ab\} \\ X_4 &= AExp \\ X_5 &= AExp \end{aligned}$$

Es gilt $f_3(X_3) = \{a+b, ab\} \not\supseteq AExp = X_4$, also $X_4 := AExp \cap \{a+b, ab\}$.
Die Kante $(4,5)$ ist noch in der Worklist enthalten.

Effizientere Fixpunktberechnung

Available-Expressions-Analyse am Beispielprogramm mittels Worklist:

Nach Iteration 3:

$$W = (4,5).(5,3)$$

$$\begin{aligned}
X_1 &= \emptyset \\
X_2 &= \{a+b\} \\
X_3 &= \{a+b, ab\} \\
X_4 &= \{a+b, ab\} \\
X_5 &= AExp
\end{aligned}$$

Es gilt $f_4(X_4) = \emptyset \nsupseteq AExp = X_5$, also $X_5 := AExp \cap \emptyset$.
Die Kante $(5,3)$ ist noch in der Worklist enthalten.

Effizientere Fixpunktberechnung

Available-Expressions-Analyse am Beispielprogramm mittels Worklist:

Nach Iteration 4:

$$W = (5, 3)$$

$$\begin{aligned}
X_1 &= \emptyset \\
X_2 &= \{a+b\} \\
X_3 &= \{a+b, ab\} \\
X_4 &= \{a+b, ab\} \\
X_5 &= \emptyset
\end{aligned}$$

Es gilt $f_5(X_5) = \{a+b\} \not\supseteq \{a+b, ab\} = X_3$, also $X_3 := \{a+b, ab\} \cap \{a+b\}$.
Die Kante $(3, 4)$ wird der Worklist hinzugefügt.

Effizientere Fixpunktberechnung

Available-Expressions-Analyse am Beispielprogramm mittels Worklist:

Nach Iteration 5:

$$W = (3, 4)$$

$$\begin{aligned} X_1 &= \emptyset \\ X_2 &= \{a + b\} \\ X_3 &= \{a + b\} \\ X_4 &= \{a + b, ab\} \\ X_5 &= \emptyset \end{aligned}$$

Es gilt $f_3(X_3) = \{a + b\} \not\supseteq \{a + b, ab\} = X_4$, also $X_4 := \{a + b, ab\} \cap \{a + b\}$.
Die Kante $(4, 5)$ wird der Worklist hinzugefügt.

Effizientere Fixpunktberechnung

Available-Expressions-Analyse am Beispielprogramm mittels Worklist:

Nach Iteration 6:

$$W = (4, 5)$$

$$\begin{aligned} X_1 &= \emptyset \\ X_2 &= \{a + b\} \\ X_3 &= \{a + b\} \\ X_4 &= \{a + b\} \\ X_5 &= \emptyset \end{aligned}$$

Es gilt $f_4(X_4) = \emptyset \supseteq \emptyset = X_5$.
Außerdem ist die Worklist nun leer.

Damit terminiert der Algorithmus.