
Programmanalyse

...speziell *Datenflussanalyse*

Üblich ist...

- die Repräsentation von Programmen durch (nichtdeterministische) *Flussgraphen*

Flussgraph

Ein (nichtdeterministischer) *Flussgraph* ist ein Quadrupel $G = (N, E, s, e)$ mit

- Knotenmenge (engl. *Nodes*) N
- Kantenmenge (engl. *Edges*) $E \subset N \times N$
- ausgezeichnetem Startknoten s ohne Vorgänger und
- ausgezeichnetem Endknoten e ohne Nachfolger

Knoten repräsentieren Programmpunkte, Kanten repräsentieren sowohl die elementaren Programmanweisungen (Zuweisungen, Tests) als auch die Verzweigungsstruktur.

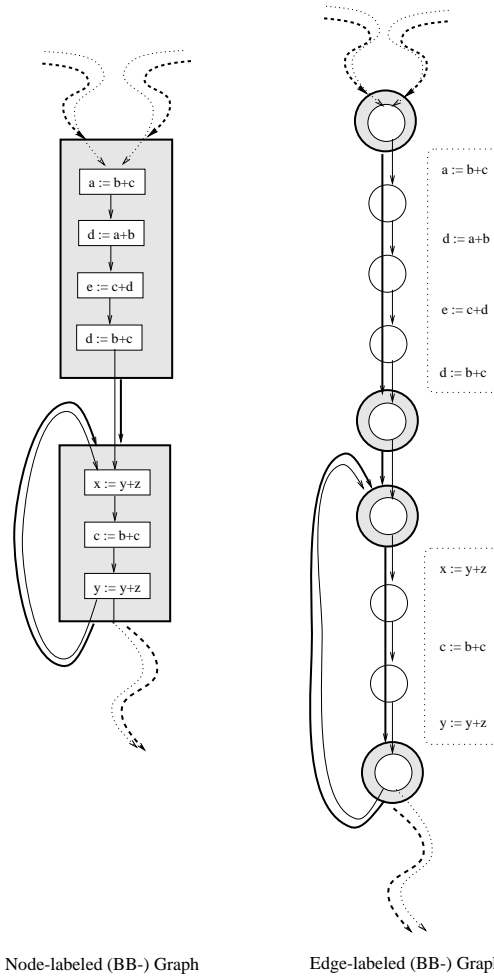
Flussgraph (2)

Flussgraphvarianten...

- Knotenbenannte Graphen
 - Einzelanweisungsgraphen (SI-Graphen)
 - Basisblockgraphen (BB-Graphen)
- Kantenbenannte Graphen
 - Einzelanweisungsgraphen (SI-Graphen)
 - Basisblockgraphen (BB-Graphen)

In der Folge werden wir bevorzugt kantenbenannte SI-Graphen betrachten.

Knoten- vs. kantenbenannte Flussgraphen



Lokale abstrakte Semantik

- *(Lokale) abstrakte Semantik*
 1. Ein *Datenflussanalyseverband* $\hat{\mathcal{C}} = (\mathcal{C}, \sqcap, \sqcup, \sqsubseteq, \perp, \top)$
 2. Ein *Datenflussanalysefunktional* $\llbracket \cdot \rrbracket : E \rightarrow (\mathcal{C} \rightarrow \mathcal{C})$

Globalisierung einer lokalen abstrakten Semantik

Zwei Strategien:

- “Meet over all Paths”-Ansatz (*MOP*)
- Maximaler Fixpunktansatz (*MaxFP*)

Der *MOP* -Ansatz

Zentral: Ausdehnung der lokalen abstrakten Semantik auf Pfade

$$\llbracket p \rrbracket =_{df} \begin{cases} Id_{\mathcal{C}} & \text{if } q < 1 \\ \llbracket \langle e_2, \dots, e_q \rangle \rrbracket \circ \llbracket e_1 \rrbracket & \text{otherwise} \end{cases}$$

Die *MOP* -Lösung

$$\forall c_s \in \mathcal{C} \ \forall n \in \mathbb{N}. \text{MOP}_{c_s}(n) = \bigsqcap \{ \llbracket p \rrbracket(c_s) \mid p \in \mathbf{P}[s, n] \}$$

Der *MaxFP* -Ansatz

Zentral: Das *MaxFP* -Gleichungssystem:

$$\mathbf{inf} (n) = \begin{cases} c_s & \text{if } n = s \\ \sqcap \{ \llbracket (m, n) \rrbracket (\mathbf{inf} (m)) \mid m \in \text{pred}(n) \} & \text{otherwise} \end{cases}$$

Die *MaxFP* -Lösung

$$\forall c_s \in \mathcal{C} \ \forall n \in \mathbb{N}. \text{MaxFP}_{(\llbracket \cdot \rrbracket, c_s)}(n) =_{df} \mathbf{inf}_{c_s}^*(n)$$

wobei $\mathbf{inf}_{c_s}^*$ die größte Lösung des *MaxFP* -Gleichungssystems bezeichnet.

Generischer Fixpunktalgorithmus 1(2)

Eingabe: (1) Ein Flussgraph $G = (N, E, s, e)$, (2) eine (lokale) abstrakte Semantik bestehend aus einem Datenflussanalyseverband \mathcal{C} , einem Datenflussanalysefunktional $\llbracket \cdot \rrbracket : E \rightarrow (\mathcal{C} \rightarrow \mathcal{C})$, und (3) einer Anfangsinformation $c_s \in \mathcal{C}$.

Ausgabe: Unter den Voraussetzungen des Effektivitätstheorems (später!) die *MaxFP*-solution. Abhängig von den Eigenschaften des Datenflussanalysefunktionals gilt dann:

(1) $\llbracket \cdot \rrbracket$ ist *distributiv*: *Variable inf* enthält für jeden Knoten die stärkste Nachbedingung bezüglich der Anfangsinformation c_s .

(2) $\llbracket \cdot \rrbracket$ ist *monoton*: *Variable inf* enthält für jeden Knoten eine sichere (d.h. untere) Approximation der stärksten Nachbedingung bezüglich der Anfangsinformation c_s .

Bemerkung: Die Variable *workset* steuert den iterativen Prozess. Ihre Elemente sind Knoten aus G , deren Annotation jüngst aktualisiert worden ist.

Generischer Fixpunktalgorithmus 2(2)

(Prolog: Initialisierung von inf and $workset$)

FORALL $n \in N \setminus \{s\}$ DO $inf[n] := \top$ OD;

$inf[s] := c_s$;

$workset := \{s\}$;

(Hauptprozess: Iterative Fixpunktberechnung)

WHILE $workset \neq \emptyset$ DO

 CHOOSE $m \in workset$;

$workset := workset \setminus \{m\}$;

 (Update the successor-environment of node m)

 FORALL $n \in succ(m)$ DO

$meet := \llbracket (m, n) \rrbracket (inf[m]) \sqcap inf[n]$;

 IF $inf[n] \sqsupset meet$

 THEN

$inf[n] := meet$;

$workset := workset \cup \{n\}$

 FI

 OD

ESOOHC

OD

Hauptresultate

Zusammenhang von...

- *MOP* - und *MaxFP* -Lösung
 - Korrektheit
 - Vollständigkeit
- *MaxFP* -Lösung und generischem Algorithmus
 - Terminierung mit *MaxFP* -Lösung

Effektivität

Theorem [Effektivität]

Der generische Fixpunktalgorithmus terminiert mit der *MaxFP*-Lösung, falls das Datenflussanalysefunktional monoton ist und der Verband die absteigende Kettenbedingung erfüllt.

Korrektheit: Sicherheitstheorem

Theorem [Sicherheit (Safety)]

Die *MaxFP*-Lösung ist eine untere Approximation der *MOP*-Lösung, d.h.,

$$\forall c_S \in \mathcal{C} \ \forall n \in \mathbb{N}. \text{MaxFP}_{c_S}(n) \sqsubseteq \text{MOP}_{c_S}(n)$$

falls das Datenflussanalysefunktional $\llbracket \cdot \rrbracket$ monoton ist.

Korrektheit und Vollständigkeit: Koinzidenztheorem

Theorem [Koinzidenz (Coincidence)]

Die *MaxFP*-solution stimmt mit der *MOP*-Lösung überein, d.h.,

$$\forall c_s \in \mathcal{C} \ \forall n \in \mathbb{N}. \text{MaxFP}_{c_s}(n) = \text{MOP}_{c_s}(n)$$

falls das Datenflussanalysefunktional $\llbracket \cdot \rrbracket$ distributiv ist.

Auf-/absteigende Kettenbedingung

Definition [Auf-/absteigende Kettenbedingung]

Ein Verband $\hat{\mathcal{C}} = (\mathcal{C}, \sqcap, \sqcup, \sqsubseteq, \perp, \top)$ erfüllt

1. die *aufsteigende Kettenbedingung*, falls jede aufsteigende Kette stationär wird, d.h. für jede Kette $p_1 \sqsubseteq p_2 \sqsubseteq \dots \sqsubseteq p_n \sqsubseteq \dots$ gibt es einen Index $m \geq 1$ so dass $x_m = x_{m+j}$ für alle $j \in \mathbb{N}$ gilt
2. die *absteigende Kettenbedingung*, falls jede absteigende Kette stationär wird, d.h. für jede Kette $p_1 \sqsupseteq p_2 \sqsupseteq \dots \sqsupseteq p_n \sqsupseteq \dots$ gibt es einen Index $m \geq 1$ so dass $x_m = x_{m+j}$ für alle $j \in \mathbb{N}$ gilt

Monotonie, Distributivität, Additivität

Definition [Monotonie, Distributivität, Additivität]

Sei $\hat{\mathcal{C}} = (\mathcal{C}, \sqcap, \sqcup, \sqsubseteq, \perp, \top)$ ein vollständiger Verband und $f : \mathcal{C} \rightarrow \mathcal{C}$ eine Funktion auf \mathcal{C} . Dann heißt f

1. *monoton* gdw $\forall c, c' \in \mathcal{C}. c \sqsubseteq c' \Rightarrow f(c) \sqsubseteq f(c')$
2. *distributiv* gdw $\forall C' \subseteq \mathcal{C}. f(\sqcap C') = \sqcap \{f(c) \mid c \in C'\}$
3. *additiv* gdw $\forall C' \subseteq \mathcal{C}. f(\sqcup C') = \sqcup \{f(c) \mid c \in C'\}$

Beispiel: Verfügbare Ausdrücke

Ein typisches distributives DFA-Problem...

- **Abstrakte Semantik für verfügbare Ausdrücke:**

1. *Datenflussanalyseverband:*

$$(\mathcal{C}, \sqcap, \sqcup, \sqsubseteq, \perp, \top) =_{df} (\mathcal{B}_X, \wedge, \vee, \leq, false, failure)$$

2. *Data-flow functional:* $\llbracket \cdot \rrbracket_{av} : E \rightarrow (\mathcal{B}_X \rightarrow \mathcal{B}_X)$ defined by

$$\forall e \in E. \llbracket e \rrbracket_{av} =_{df} \begin{cases} Cst_{true}^X & \text{if } Comp_e \wedge Transp_e \\ Id_{\mathcal{B}_X} & \text{if } \neg Comp_e \wedge Transp_e \\ Cst_{false}^X & \text{otherwise} \end{cases}$$

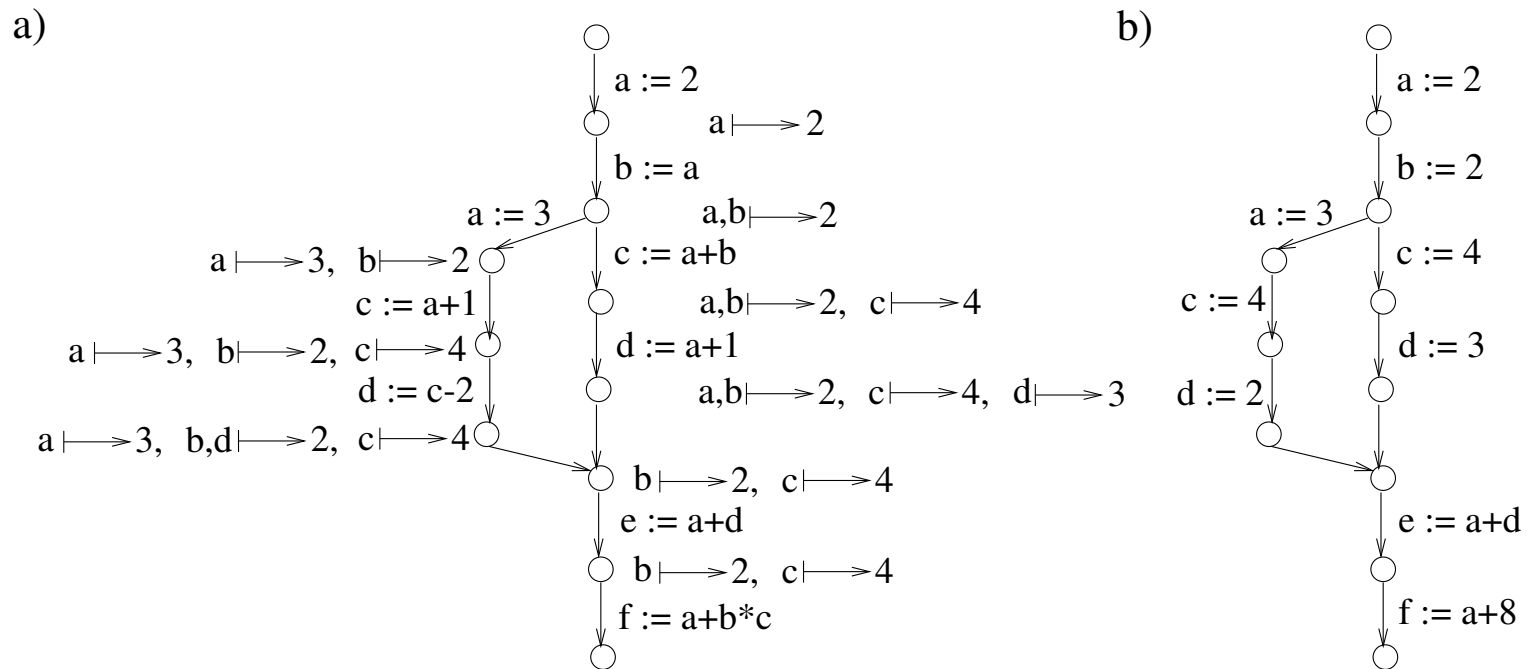
wobei

$$\hat{\mathcal{B}}_X =_{df} (\mathcal{B}_X, \wedge, \vee, \leq, false, failure)$$

mit $false \leq true \leq failure$ und dem logischen “und” und “oder” als Schnitt- bzw. Vereinigungsoperation \sqcap and \sqcup .

Beispiel: Einfache Konstanten

Ein typisches monotones DFA-Problem...



Abstrakte Semantik für einfache Konstanten

- **Abstrakte Semantik für einfache Konstanten:**

1. *Datenflussanalyseverband:*

$$(\mathcal{C}, \sqcap, \sqcup, \sqsubseteq, \perp, \top) =_{df} (\Sigma_X, \sqcap, \sqcup, \sqsubseteq, \sigma_\perp, \sigma_{failure})$$

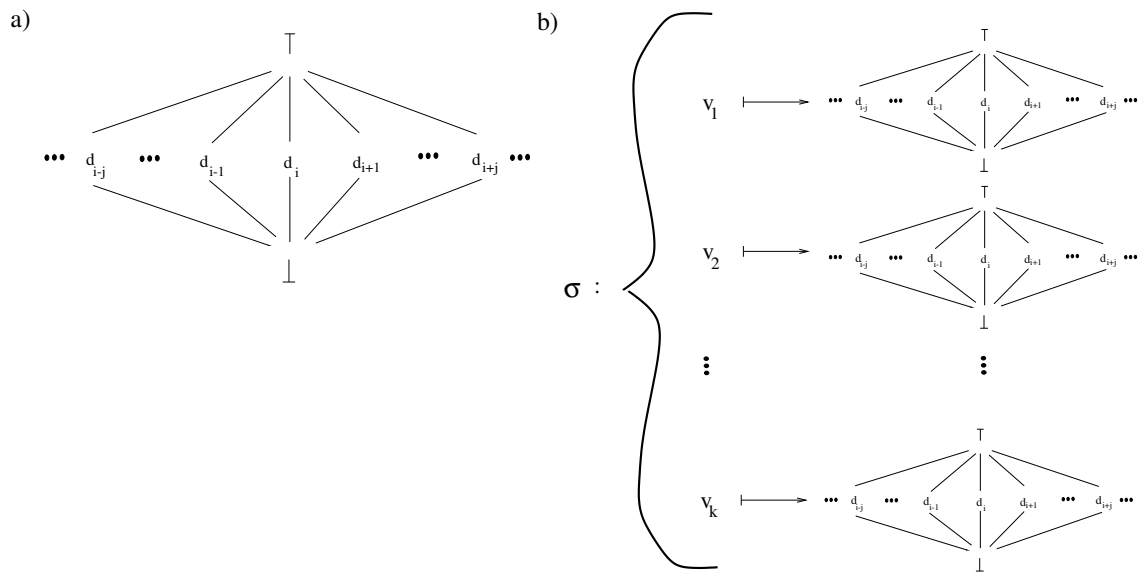
2. *Datenflussanalysefunktional:*

$\llbracket \cdot \rrbracket_{sc} : E \rightarrow (\Sigma_X \rightarrow \Sigma_X)$ definiert durch

$$\forall e \in E. \llbracket e \rrbracket_{sc} =_{df} \theta_e$$

Datenflussanalyseverband für einfache Konstanten

Der "kanonische" Verband für Konstantenausbreitung/-faltung:



Die Semantik von Termen

Die *Semantik* von Termen $t \in \mathbf{T}$ ist gegeben durch eine *Evaluationsfunktion*

$$\mathcal{E} : \mathbf{T} \rightarrow (\Sigma_X \rightarrow \mathbf{D})$$

Diese Funktion ist undefiniert für den speziellen Fehlerzustand $\sigma_{failure}$, und induktiv definiert für gewöhnliche Zustände:

$$\forall t \in \mathbf{T} \forall \sigma \in \Sigma. \mathcal{E}(t)(\sigma) =_{df} \begin{cases} \sigma(x) & \text{falls } t = x \in \mathbf{V} \\ I_0(c) & \text{falls } t = c \in \mathbf{C} \\ I_0(op)(\mathcal{E}(t_1)(\sigma), \dots, \mathcal{E}(t_r)(\sigma)) & \text{falls } t = op(t_1, \dots, t_r) \end{cases}$$

Menge der Zustände

$$\Sigma =_{df} \{ \sigma \mid \sigma : \mathbf{V} \rightarrow \mathbf{D} \}$$

bezeichnet die Menge der *gewöhnlichen Zustände*.

$$\Sigma_X =_{df} \Sigma \cup \{ \sigma_{failure} \}$$

bezeichnet die Menge aller *Zustände*, wobei $\sigma_{failure}$ einen speziellen Fehlerzustand bezeichnet.

Zustandstransformationsfunktion

Die *Zustandstransformationsfunktion*

$$\theta_\iota : \Sigma_X \rightarrow \Sigma_X, \quad \iota \equiv x := t$$

lässt den Fehlerzustand $\sigma_{failure}$ invariant und bildet alle gewöhnlichen Zustände wie folgt ab:

$$\forall \sigma \in \Sigma \quad \forall y \in \mathbf{V}. \theta_\iota(\sigma)(y) =_{df} \begin{cases} \mathcal{E}(t)(\sigma) & \text{if } y = x \\ \sigma(y) & \text{otherwise} \end{cases}$$

Der funktionale *MaxFP* -Ansatz

Zentral: Das “funktionale” *MaxFP* -Gleichungssystem:

$$\llbracket n \rrbracket = \begin{cases} Id_{\mathcal{C}} & \text{falls } n = s \\ \bigsqcap \{ \llbracket (n, m) \rrbracket \circ \llbracket m \rrbracket \mid m \in pred(n) \} & \text{sonst} \end{cases}$$

In der Folge bezeichne das Funktional

$$\llbracket \rrbracket : N \rightarrow (\mathcal{C} \rightarrow \mathcal{C})$$

die größte Lösung des obigen Gleichungssystems.

Zusammenhang von *MaxFP* - und funktionalem *MaxFP* -Ansatz

Theorem [Äquivalenz]

$$\forall n \in N \ \forall c_s \in \mathcal{C}. \text{MFP}_{(G, \llbracket \cdot \rrbracket)}(n)(c_s) = \llbracket n \rrbracket(c_s)$$

Ausblick

Die funktionale Perspektive auf den *MaxFP* -Ansatz liefert den Schlüssel zu

- interprozeduraler (d.h. von Programmen mit Prozeduren)
- paralleler (d.h. von Programmen mit Parallelität)

Datenflussanalyse.

Vorschau auf die weiteren Vorlesungstermine...

- Di, 16.01.2007, Vorlesung von 17:45 Uhr bis 19:15 Uhr, Bibliothek E185/1
- Di, 23.01.2007, Vorlesung von 17:45 Uhr bis 19:15 Uhr, Bibliothek E185/1
- Di, 30.01.2007, Vorlesung von 17:45 Uhr bis 19:15 Uhr, Bibliothek E185/1