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.

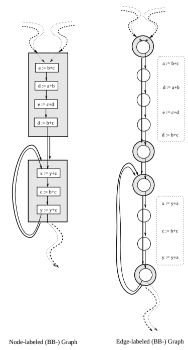
Flussgraph (2)

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* $\hat{\mathcal{C}} = (\mathcal{C}, \sqcap, \sqcup, \sqsubseteq, \bot, \top)$
  2. Ein *Datenflussanalysefunktional* $[\![\ ]\!] : E \to (\mathcal{C} \to \mathcal{C})$

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

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

Die MOP-Lösung

$$\forall\, c_{\mathbf{s}} \in \mathcal{C}\ \forall\, n \in N.\ MOP_{c_{\mathbf{s}}}(n) = \bigsqcap \{\, [\![\, p \,]\!](c_{\mathbf{s}}) \mid p \in \mathbf{P}[\mathbf{s}, n] \,\}$$

Der MaxFP -Ansatz

Zentral: Das MaxFP -Gleichungssystem:

$$\textbf{inf}\ (n) \ = \ \begin{cases} c_s & \text{if } n = s \\ \bigsqcap \{ [\![(m,n)]\!](\textbf{inf}\ (m)) \mid m \in \mathit{pred}(n) \} & \text{otherwise} \end{cases}$$

Die MaxFP -Lösung

$\forall c_s \in \mathcal{C}\ \forall n \in N.\ \mathit{MaxFP}_{([\![\ ]\!], c_s)}(n) =_{df} \textbf{inf}^{\,*}_{c_s}(n)$

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

Generischer Fixpunktalgorithmus 1(2)

Eingabe: (1) Ein Flussgraph $G = (N, E, \mathbf{s}, \mathbf{e})$, (2) eine (lokale) abstrakte Semantik bestehend aus einem Datenflussanalyseverband $\mathcal{C}$, einem Datenflussanalysefunktional $[\![\ ]\!] : E \to (\mathcal{C} \to \mathcal{C})$, 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) $[\![\ ]\!]$ ist *distributiv*: Variable *inf* enthält für jeden Knoten die stärkste Nachbedingung bezüglich der Anfangsinformation c_s.

(2) $[\![\ ]\!]$ 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 Process. Ihre Elemente sind Knoten aus G, deren Annotation jüngst aktualisiert worden ist.

Generischer Fixpunktalgorithmus 2(2)

```
( Prolog: Initialisierung von inf and workset )
FORALL n ∈ N\{s} DO inf[n] := ⊤ OD;
inf[s] := c_s;
workset := { s };

( Hauptprozess: Iterative Fixpunktberechnung )
WHILE workset ≠ ∅ DO
  CHOOSE m ∈ workset;
   workset := workset\{ m };
   ( Update the successor-environment of node m )
   FORALL n ∈ succ(m) DO
    meet := [[ (m,n) ]](inf[m]) ⊓ inf[n];
    IF inf[n] ⊐ meet
      THEN
        inf[n] := meet;
        workset := workset ∪ { 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 Fixpunktalgorithm 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.,

$$\forall c_s \in \mathcal{C}\ \forall n \in N.\ \mathit{MaxFP}_{c_s}(n) \sqsubseteq \mathit{MOP}_{c_s}(n)$$

falls das Datenflussanalysefunktional $[\![\ ]\!]$ monoton ist.

Korrektheit und Vollständigkeit: Koinzidenztheorem

Theorem [Koinzidenz (Coincidence)]

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

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

falls das Datenflussanalysefunktional $[\![\ ]\!]$ distributiv ist.

Auf-/absteigende Kettenbedingung

Definition [Auf-/absteigende Kettenbedingung]

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

1. die *aufsteigende Kettenbedingung*, falls jede aufsteigende Kette stationär wird, d.h. für jede Kette $p_1 \sqsubseteq p_2 \sqsubseteq \ldots \sqsubseteq p_n \sqsubseteq \ldots$ 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 $p_1 \sqsupseteq p_2 \sqsupseteq \ldots \sqsupseteq p_n \sqsupseteq \ldots$ 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 $\widehat{\mathcal{C}} = (\mathcal{C}, \sqcap, \sqcup, \sqsubseteq, \bot, \top)$ ein vollständiger Verband und $f : \mathcal{C} \to \mathcal{C}$ 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 $\forall\, C' \subseteq \mathcal{C}.\ \ f(\bigsqcap C') = \bigsqcap \{ f(c) \mid c \in C' \}$
3. *additiv* gdw $\forall\, C' \subseteq \mathcal{C}.\ \ f(\bigsqcup C') = \bigsqcup \{ f(c) \mid c \in C' \}$

Beispiel: Verfügbare Ausdrücke

Ein typisches distributives DFA-Problem...

- **Abstrakte Semantik für verfügbare Ausdrücke**:
  1. *Datenflussanalyseverband*:
     $(\mathcal{C}, \sqcap, \sqcup, \sqsubseteq, \bot, \top) =_{df} (\mathcal{B}_X, \wedge, \vee, \leq, \mathit{false}, \mathit{failure})$
  2. *Data-flow functional*: $[\![\ ]\!]_{av} : E \to (\mathcal{B}_X \to \mathcal{B}_X)$ defined by

$$\forall\, e \in E.\ [\![\, e \,]\!]_{av} =_{df} \begin{cases} Cst^X_{true} & \text{if } \mathit{Comp}_e \wedge \mathit{Transp}_e \\ Id_{\mathcal{B}_X} & \text{if } \neg \mathit{Comp}_e \wedge \mathit{Transp}_e \\ Cst^X_{false} & \text{otherwise} \end{cases}$$

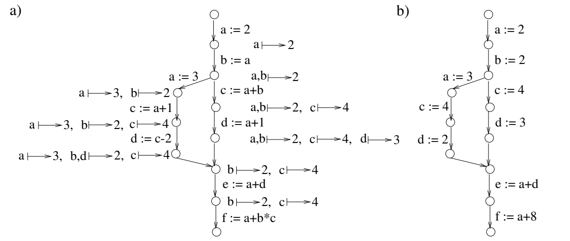
wobei

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

mit $\mathit{false} \leq \mathit{true} \leq \mathit{failure}$ und dem logischen "und" und "oder" als Schnitt- bzw. Vereinigungsoperation $\sqcap$ and $\sqcup$.

Beispiel: Einfache Konstanten

Ein typisches monotones DFA-Problem...



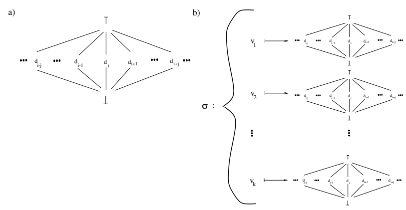
Abstrakte Semantik für einfache Konstanten

- **Abstrakte Semantik für einfache Konstanten**:
  1. *Datenflussanalyseverband*:
     $(\mathcal{C}, \sqcap, \sqcup, \sqsubseteq, \bot, \top) =_{df} (\Sigma_X, \sqcap, \sqcup, \sqsubseteq, \sigma_\bot, \sigma_{failure})$
  2. *Datenflussanalysefunktional*:
     $[\![\ ]\!]_{sc} : E \to (\Sigma_X \to \Sigma_X)$ definiert durch

$$\forall\, e \in E.\ [\![\, e \,]\!]_{sc} =_{df} \theta_e$$

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*

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

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

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

Menge der Zustände

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

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*

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

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

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

Der funktionale $MaxFP$ -Ansatz

Zentral: Das "funktionale" $MaxFP$ -Gleichungssystem:

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

In der Folge bezeichne das Funktional

$$[\![\ ]\!] : N \to (\mathcal{C} \to \mathcal{C})$$

die größte Lösung des obigen Gleichungssytems.

Zusammenhang von $MaxFP$ - und funktionalem $MaxFP$ -Ansatz

Theorem [Äquivalenz]

$\forall\, n \in N\ \forall\, c_{\mathbf{s}} \in \mathcal{C}.\ MFP_{(G,[\![\ ]\!])}(n)(c_s) = [\![\, n \,]\!](c_{\mathbf{s}})$

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