Programmanalyse

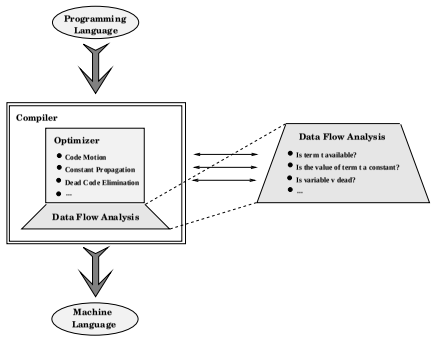
...speziell *Datenflussanalyse*

Typische Fragen sind...

- Welchen *Wert* hat eine Variable an einer Programmstelle?
  $\leadsto$ Konstantenausbreitung und Faltung
- Steht der Wert eines Ausdrucks an einer Programmstelle *verfügbar*?
  $\leadsto$ (Partielle) Redundanzelimination
- Ist eine Variable *tot* an einer Programmstelle?
  $\leadsto$ Elimination (partiell) toten Codes

Hintergrund

...(Programm-) Analyse zur (Programm-) Optimierung



In der Folge

Zentrale Fragen...

Grundlegendes ebenso...

- Was heißt *Optimalität*
  ...in Analyse und in Optimierung?

...wie (scheinbar) Nebensächliches:

- Was ist eine *angemessene* Programmrepräsentation?

Ausblick

Genauer werden wir unterscheiden...

- Intraprozedurale,
- interprozedurale,
- parallele,
- konditionale,
- ...

Datenflussanalyse.

Ausblick

Ingredienzien *intraprozeduraler* Datenflussanalyse:

- *(Lokale) abstrakte Semantik*
  1. Ein *Datenflussanalyseverband* $\hat{\mathcal{C}} = (\mathcal{C}, \sqcap, \sqcup, \sqsubseteq, \bot, \top)$
  2. Ein *Datenflussanalysefunktional* $[\![\ ]\!] : E \to (\mathcal{C} \to \mathcal{C})$
  3. Anfangsinformation/-zusicherung $c_{\mathbf{s}} \in \mathcal{C}$
- *Globalisierungsstrategien*
  1. "Meet over all Paths"-Ansatz (MOP)
  2. Maximaler Fixpunktansatz ($MaxFP$)
- *Generischer Fixpunktalgorithmus*

Ausblick

Hauptresultate:

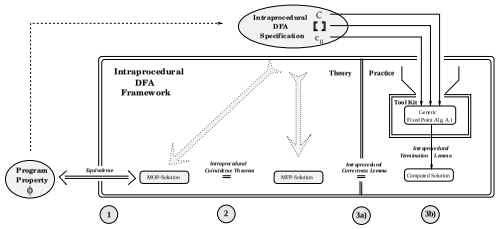
- *Sicherheits-* (*Korrektheits-*) Theorem
- *Koinzidenz-* (*Vollständigkeits-*) Theorem

Sowie:

- *Effektivitäts-* (*Terminierungs-*) Theorem

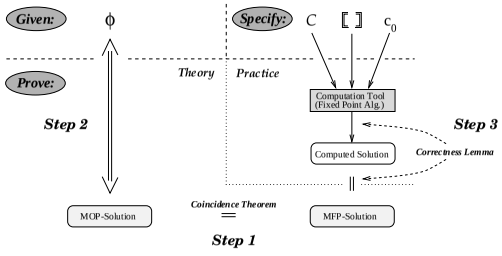
Ausblick: Intraprozedurale Datenfluss-analyse (1)

...die (detaillierte) Werkzeugkistensicht:



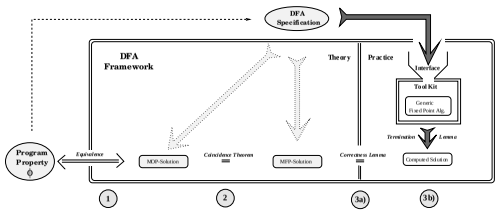
Ausblick: Intraprozedurale Datenfluss-analyse (2)

...bei genauerem Hineinsehen:



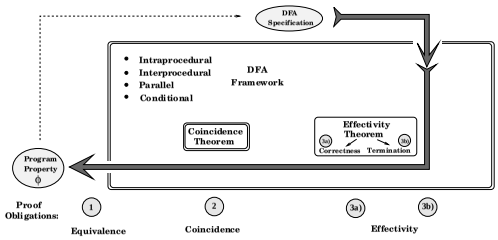
Ausblick: DFA-Frameworks / DFA-Toolkits (1)

...aus größerer Ferne und Konzentration auf das Wesentliche:



Ausblick: DFA-Frameworks / DFA-Toolkits (2)

...das generelle Muster, die Werkzeugkistensicht:



Ziel

Optimale Programmoptimierung...

...*weiße Schimmel* in der Informatik?

Ohne Fleiß kein Preis!

In der Sprechweise der optimierenden Übersetzung...

...ohne *Analyse* keine Optimierung!

Zurück zum Anfang: Zur 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

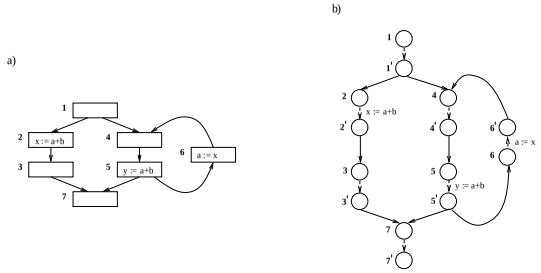
- Knotenmenge (engl. *Nodes*) N
- Kantenmenge (engl. *Edges*) $E \subseteq N \times N$
- ausgezeichnetem Startknoten s ohne Vorgänger und
- ausgezeichnetem Endknoten e ohne Nachfolger

Knoten repräsentieren Programmpunkte, Kanten repräsentieren die Verzweigungsstruktur. Elementare Programmanweisungen (Zuweisungen, Tests) können wahlweise durch Knoten oder Kanten repräsentiert werden.

$\rightsquigarrow$ Darstellungsvarianten: Knoten- vs. kantenbenannte Flussgraphen

Veranschaulichung

Knoten- vs. kantenbenannte Flussgraphen
(hier mit Einzelanweisungsbenennung)



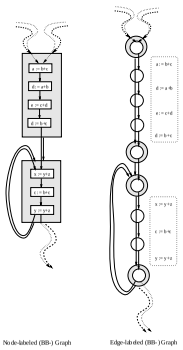
Flussgraphen

Darstellungsvarianten...

- 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



Bezeichnungen

Sei $G = (N, E, s, e)$ ein Flussgraph, seien m, n zwei Knoten aus N. Dann bezeichne:

- $\mathbf{P}_G[m, n]$: ...die Menge aller Pfade von m nach n
- $\mathbf{P}_G[m, n[$: ...die Menge aller Pfade von m zu einem Vorgänger von n
- $\mathbf{P}_G]m, n]$: ...die Menge aller Pfade von einem Nachfolger von m nach n
- $\mathbf{P}_G]m, n[$: ...die Menge aller Pfade von einem Nachfolger von m zu einem Vorgänger von n

Bem.: Wenn G aus dem Kontext eindeutig hervorgeht, schreiben wir einfacher auch $\mathbf{P}$ statt $\mathbf{P}_G$.

Datenflussanalysespezifikation

- *(Lokale) abstrakte Semantik*
  1. Ein *Datenflussanalyseverband* $\hat{\mathcal{C}} = (\mathcal{C}, \sqcap, \sqcup, \sqsubseteq, \bot, \top)$
  2. Ein *Datenflussanalysefunktional* $[\![\ ]\!] : E \to (\mathcal{C} \to \mathcal{C})$
- Eine Anfangsinformation/-zusicherung: $c_{\mathbf{s}} \in \mathcal{C}$

Globalisierung einer lokalen abstrakten Semantik

Zwei Strategien:

- "Meet over all Paths"-Ansatz (MOP)
  $\leadsto$ *spezifizierende* Lösung
- Maximaler Fixpunktansatz ($MaxFP$)
  $\leadsto$ *berechenbare* Lösung

Der MOP -Ansatz

Zentral: Ausdehnung der lokalen abstrakten Semantik auf Pfade

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

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:

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

Die $MaxFP$ -Lösung

$$\forall\, c_{\mathbf{s}} \in \mathcal{C}\ \forall\, n \in N.\ MaxFP_{([\![\ ]\!], c_{\mathbf{s}})}(n) =_{df} \mathbf{inf}^{\,*}_{c_{\mathbf{s}}}(n)$$

wobei $\mathbf{inf}^{\,*}_{c_{\mathbf{s}}}$ 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 $c_{\mathbf{s}} \in \mathcal{C}$.

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 $c_{\mathbf{s}}$.

(2) $[\![\ ]\!]$ ist *monoton*: Variable *inf* enthält für jeden Knoten eine sichere (d.h. untere) Approximation der stärksten Nachbedingung bezüglich der Anfangsinformation $c_{\mathbf{s}}$.

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 $n \in N \backslash \{\mathbf{s}\}$ DO *inf*$[n] := \top$ OD;
inf$[\mathbf{s}] := c_{\mathbf{s}}$;
workset $:= \{\mathbf{s}\}$;

(Hauptprozess: Iterative Fixpunktberechnung)
WHILE *workset* $\neq \emptyset$ DO
  CHOOSE $m \in$ *workset*;
    workset $:=$ *workset* $\backslash \{m\}$;
    (Aktualisiere die Nachfolgerumgebung von Knoten m)
    FORALL $n \in succ(m)$ DO
      meet $:= [\![(m, n)]\!]$(*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

Korrektheit: Sicherheitstheorem

Theorem [Sicherheit (Safety)]

Die $MaxFP$-Lösung ist eine sichere (konservative). d.h. untere Approximation der MOP-Lösung, d.h.,

$$\forall\, c_{\mathbf{s}} \in \mathcal{C}\ \forall\, n \in N.\ MaxFP_{c_{\mathbf{s}}}(n) \sqsubseteq MOP_{c_{\mathbf{s}}}(n)$$

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

Vollständigkeit (und Korrektheit): Koinzidenztheorem

Theorem [Koinzidenz (Coincidence)]

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

$$\forall\, c_{\mathbf{s}} \in \mathcal{C}\ \forall\, n \in N.\ MaxFP_{c_{\mathbf{s}}}(n) = MOP_{c_{\mathbf{s}}}(n)$$

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

Terminierung: Effektivitätstheorem

Theorem [Effektivität]

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

Nachzutragende Definitionen

...sind:

- Absteigende (aufsteigende) Kettenbedingung
- Monotonie und Distributivität von Datenflussanalysefunktionalen

Auf-/absteigende Kettenbedingung

Definition [Ab-/aufsteigende Kettenbedingung]

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

1. 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
2. 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

Monotonie, Distributivität, Additivität

...von Funktionen auf (Datenflussanalyse-) Verbänden.

Definition [Monotonie, Distributivität, Additivität]
Sei $\hat{\mathcal{C}} = (\mathcal{C}, \sqcap, \sqcup, \sqsubseteq, \bot, \top)$ ein vollständiger Verband und $f : \mathcal{C} \to \mathcal{C}$ eine Funktion auf $\mathcal{C}$. Dann heißt f

1. *monoton* gdw $\forall\, c, c' \in \mathcal{C}.\ c \sqsubseteq c' \Rightarrow f(c) \sqsubseteq f(c')$
   (Erhalt der Ordnung der Elemente)
2. distributiv gdw $\forall\, C' \subseteq \mathcal{C}.\ f(\bigsqcap C') = \bigsqcap \{f(c) \mid c \in C'\}$
   (Erhalt der größten unteren Schranken)
3. *additiv* gdw $\forall\, C' \subseteq \mathcal{C}.\ f(\bigsqcup C') = \bigsqcup \{f(c) \mid c \in C'\}$
   (Erhalt der kleinsten oberen Schranken)

Zur Erinnerung: Oft nützlich

...ist folgende äquivalente Charakterisierung der Monotonie:

Lemma

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

$$f \text{ ist monoton} \iff \forall\, C' \subseteq \mathcal{C}.\ f(\bigsqcap C') \sqsubseteq \bigsqcap \{f(c) \mid c \in C'\}$$

Monotonie und Distributivität

...von Datenflussanalysefunktionalen.

Definition
Ein Datenflussanalysefunktional $[\![\ ]\!] : E \to (\mathcal{C} \to \mathcal{C})$ heißt *monoton* (*distributiv*) gdw $\forall\, e \in E.\ [\![e]\!]$ ist monoton (distributiv).

Beispiel: Verfügbare Ausdrücke

...ein typisches distributives DFA-Problem.

- **Abstrakte Semantik für verfügbare Ausdrücke**:
  1. *Datenflussanalyseverband*:
     $(\mathcal{C}, \sqcap, \sqcup, \sqsubseteq, \bot, \top) =_{df} (\mathcal{B}, \wedge, \vee, \leq, \mathit{false}, \mathit{true})$
  2. *Datenflussanalysefunktional*: $[\![\ ]\!]_{av} : E \to (\mathcal{B} \to \mathcal{B})$ definiert durch

$$\forall\, e \in E.\ [\![e]\!]_{av} =_{df} \begin{cases} Cst_{true} & \text{falls } \mathit{Comp}_e \wedge \mathit{Transp}_e \\ Id_{\mathcal{B}} & \text{falls } \neg \mathit{Comp}_e \wedge \mathit{Transp}_e \\ Cst_{false} & \text{sonst} \end{cases}$$

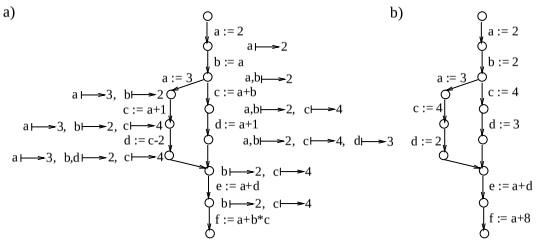
wobei

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

den Verband der Wahrheitswerte bezeichnet mit $\mathit{false} \leq \mathit{true}$ und dem logischen "und" und "oder" als Schnitt- bzw. Vereinigungsoperation $\sqcap$ and $\sqcup$.

Beispiel: Einfache Konstanten

Ein typisches monotones (nicht distributives) DFA-Problem...



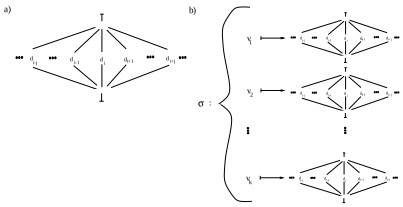
Abstrakte Semantik für einfache Konstanten

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

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

die induktiv definiert ist durch:

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

Nachzutragende Begriffe und Definitionen

...um die Definition der Termsemantik abzuschließen:

- Termsyntax
- Interpretation
- Zustand

Die Syntax von Termen (1)

Sei

- $\mathbf{V}$ eine Menge von Variablen und
- $\mathbf{Op}$ eine Menge von n-stelligen Operatoren, $n \geq 0$, sowie $\mathbf{C} \subseteq \mathbf{Op}$ die Menge der 0-stelligen Operatoren, der sog. *Konstanten* in $\mathbf{Op}$.

Die Syntax von Termen (2)

Dann legen wir fest:

1. Jede Variable $v \in \mathbf{V}$ und jede Konstante $c \in \mathbf{C}$ ist ein Term.
2. Ist $op \in \mathbf{Op}$ ein n-stelliger Operator, $n \geq 1$, und sind $t_1, \ldots, t_n$ Terme, dann ist auch $op(t_1, \ldots, t_n)$ ein Term.
3. Es gibt keine weiteren Terme außer den nach den obigen beiden Regeln konstruierbaren.

Die Menge aller Terme bezeichnen wir mit $\mathbf{T}$.

Interpretation

Sei $\mathbf{D}'$ ein geeigneter Datenbereich (z.B. die Menge der ganzen Zahlen), seien $\bot$ und $\top$ zwei ausgezeichnete Elemente mit $\bot, \top \notin \mathbf{D}'$ und sei $\mathbf{D} =_{df} \mathbf{D}' \cup \{\bot, \top\}$.

Eine *Interpretation* über $\mathbf{T}$ und $\mathbf{D}$ ist ein Paar $I \equiv (\mathbf{D}, I_0)$, wobei

- I_0 eine Funktion ist, die mit jedem 0-stelligen Operator $c \in \mathbf{Op}$ ein Datum $I_0(c) \in \mathbf{D}'$ und mit jedem n-stelligen Operator $op \in \mathbf{Op}$, $n \geq 1$, eine totale Funktion $I_0(op) : \mathbf{D}^n \to \mathbf{D}$ assoziiert, die als *strikt* angenommen wird (d.h. $I_0(op)(d_1, \ldots, d_n) = \bot$, wann immer es ein $j \in \{1, \ldots, n\}$ gibt mit $d_j = \bot$)

Menge der Zustände

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

...bezeichnet die Menge der *Zustände*, d.h. die Menge der Abbildungen σ von der Menge der Programmvariablen $\mathbf{V}$ auf einen geeigneten (hier nicht näher spezifizierten) Datenbereich $\mathbf{D}$.

Insbesondere

- $\sigma_\bot$: ...bezeichnet den wie folgt definierten *total undefinierten* Zustand aus Σ: $\forall\, v \in \mathbf{V}.\ \sigma_\bot(v) = \bot$

Zustandstransformationsfunktion

Die *Zustandstransformationsfunktion*

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

ist definiert durch:

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

Der funktionale $MaxFP$ -Ansatz

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

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

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]

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

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...

- Mo, 26.11.2007: Vorlesung von 16:15 Uhr bis 17:45 Uhr im Hörsaal 14, TU-Hauptgebäude
- Mo, 03.12.2007: Vorlesung von 16:15 Uhr bis 17:45 Uhr im Hörsaal 14, TU-Hauptgebäude
- Mo, 10.12.2007: Vorlesung von 16:15 Uhr bis 17:45 Uhr im Hörsaal 14, TU-Hauptgebäude
- *Mo, 17.12./24.12./31.12.2007: Keine Vorlesung(en)!* (*Ferialzeit*)