TECHNISCHE UNIVERSITÄT DRESDEN

ALGORITHMEN UND DATENSTRUKTUREN

ÜBUNG 15: WIEDERHOLUNG & KONSULTATION

Eric Kunze

eric.kunze@tu-dresden.de

TU Dresden, 31.01.2022

letzte Änderung:
29.01.2022, 21:51

Fixpunktsemantik

EBNF-DEFINITION

$$\mathcal{E} = (\underbrace{V}_{\text{Nichtterminalsymbole}}, \underbrace{\Sigma}_{\text{Terminalsymbole}}, \underbrace{S}_{\text{Startsymbol}}, \underbrace{R}_{\text{Regelmenge}})$$

Jede **EBNF-Regel** besteht aus einer linken und einer rechten Seite, die rechte Seite ist ein **EBNF-Term**.

Nichtterminalsymbol ::= *EBNF-Term*

EBNF-DEFINITION

$$\mathcal{E} = (\ V,\ \Sigma,\ S,\ R\ )$$

Nichtterminalsymbole

Terminalsymbole

Startsymbol

Regelmenge

Jede **EBNF-Regel** besteht aus einer linken und einer rechten Seite, die rechte Seite ist ein **EBNF-Term**.

Nichtterminalsymbol ::= *EBNF-Term*

Definition (EBNF-Terme): Seien V (syntaktische Variablen) und Σ (Terminalsymbole) endliche Mengen mit $V \cap \Sigma = \varnothing$. Die Menge der EBNF-Terme über V und Σ (notiere: $T(\Sigma, V)$), ist die *kleinste* Menge $T \subseteq \left(V \cup \Sigma \cup \left\{\hat{\{}, \hat{\}}, \hat{[}, \hat{]}, \hat{(}, \hat{)}, \hat{|}\right\}\right)$ mit $V \subseteq T$, $\Sigma \subseteq T$ und

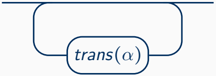
- Wenn $\alpha \in T$, so auch $\hat{(}\ \alpha\ \hat{)} \in T$, $\hat{\{}\ \alpha\ \hat{\}} \in T$, $\hat{[}\ \alpha\ \hat{]} \in T$.
- Wenn $\alpha_1, \alpha_2 \in T$, so auch $\hat{(}\ \alpha_1\ \hat{|}\ \alpha_2\ \hat{)} \in T$, $\alpha_1\alpha_2 \in T$.

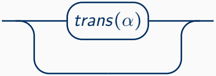
ÜBERSETZUNG EBNF ↔ SYNTAXDIAGRAMME

Sei $v \in V$ und $w \in \Sigma$. $\mathit{trans}(v) =$ 

; $\mathit{trans}(w) =$ 

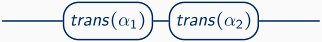
Sei $\alpha \in T(\Sigma, V)$ ein EBNF-Term.

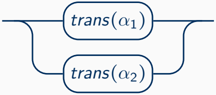
- $\mathit{trans}(\ \hat{\{}\ \alpha\ \hat{\}}\ ) =$ 

- $\mathit{trans}(\ \hat{[}\ \alpha\ \hat{]}\ ) =$ 

- $\mathit{trans}(\ \hat{(}\ \alpha\ \hat{)}\ ) = \mathit{trans}(\alpha)$

Seien $\alpha_1, \alpha_2 \in T(\Sigma, V)$ zwei EBNF-Terme.

- $\mathit{trans}(\ \alpha_1\alpha_2\ ) =$ 

- $\mathit{trans}(\ \hat{(}\ \alpha_1\ \hat{|}\ \alpha_2\ \hat{)}\ ) =$ 

Auszug aus dem Foliensatz der Vorlesung

SEMANTIK VON EBNF-TERMEN

Ziel: Ordne einer EBNF-Definition $\mathcal{E} = (V, \Sigma, S, R)$ ihre Sprache zu

- $W(\mathcal{E}, v)$ bezeichnet von $v \in V$ beschriebene Objektsprache
- $\rho \colon V \to \mathcal{P}(\Sigma^*)$ ordnet jeder syntaktischen Variable $v \in V$ eine Sprache zu
- Vorstellung: $\rho(v)$ ist bestes Wissen über die von v beschriebene Sprache

SEMANTIK VON EBNF-TERMEN

Ziel: Ordne einer EBNF-Definition $\mathcal{E} = (V, \Sigma, S, R)$ ihre Sprache zu

- $W(\mathcal{E}, v)$ bezeichnet von $v \in V$ beschriebene Objektsprache
- $\rho\colon V \to \mathcal{P}(\Sigma^*)$ ordnet jeder syntaktischen Variable $v \in V$ eine Sprache zu
- Vorstellung: $\rho(v)$ ist bestes Wissen über die von v beschriebene Sprache

Problem: Wie bekomme ich aus einem EBNF-Term eine Sprache?

Semantik $[\![\cdot]\!]: \underbrace{T(\Sigma, V)}_{\text{EBNF-Term } \alpha} \to ((\underbrace{V \to \mathcal{P}(\Sigma^*)}_{\rho}) \to \mathcal{P}(\Sigma^*))$

SEMANTIK VON EBNF-TERMEN

$$[\![\cdot]\!] : \underbrace{T(\Sigma, V)}_{\text{EBNF-Term } \alpha} \to ((\underbrace{V \to \mathcal{P}(\Sigma^*)}_{\rho}) \to \mathcal{P}(\Sigma^*))$$

Sei $\alpha \in T(\Sigma, V)$ ein EBNF-Term. Die Semantik $[\![\alpha]\!](\rho)$ von α ist definiert als:

- Wenn $\alpha = v \in V$, dann gilt $[\![\alpha]\!](\rho) = \rho(v)$.
- Wenn $\alpha = w \in \Sigma$, dann gilt $[\![\alpha]\!](\rho) = \{w\}$.
- Wenn $\alpha = \hat{\{}\ \alpha_1\ \hat{\}}$, dann gilt $[\![\alpha]\!](\rho) = ([\![\alpha_1]\!](\rho))^*$.
- Wenn $\alpha = \hat{[}\ \alpha_1\ \hat{]}$, dann gilt $[\![\alpha]\!](\rho) = [\![\alpha_1]\!](\rho) \cup \{\varepsilon\}$.
- Wenn $\alpha = \hat{(}\ \alpha_1\ \hat{)}$, dann gilt $[\![\alpha]\!](\rho) = [\![\alpha_1]\!](\rho)$.
- Wenn $\alpha = \alpha_1\alpha_2$, dann gilt $[\![\alpha]\!](\rho) = [\![\alpha_1]\!](\rho) \cdot [\![\alpha_2]\!](\rho)$.
- Wenn $\alpha = \hat{(}\ \alpha_1\ \hat{|}\ \alpha_2\ \hat{)}$, dann gilt $[\![\alpha]\!](\rho) = [\![\alpha_1]\!](\rho) \cup [\![\alpha_2]\!](\rho)$.

FIXPUNKTITERATION – EINE ANALOGIE

Ausblick: Fixpunktiteration zur Nullstellenbestimmung

Gegeben sei eine Funktion $g: \mathbb{R} \to \mathbb{R}$, von der wir eine Nullstelle suchen, d.h. ein $\overline{x} \in \mathbb{R}$ mit $g(\overline{x}) = 0$.

FIXPUNKTITERATION – EINE ANALOGIE

Ausblick: Fixpunktiteration zur Nullstellenbestimmung

Gegeben sei eine Funktion $g\colon \mathbb{R} \to \mathbb{R}$, von der wir eine Nullstelle suchen, d.h. ein $\overline{x} \in \mathbb{R}$ mit $g(\overline{x}) = 0$.

Methode: Newtonverfahren — definiere $\Phi(x) := x - \frac{g(x)}{g'(x)}$.

- Starte mit „beliebigem“ Startwert $x_0 \in \mathbb{R}$.
- Berechne stets $x_{i+1} = \Phi(x_i)$.

FIXPUNKTITERATION – EINE ANALOGIE

Ausblick: Fixpunktiteration zur Nullstellenbestimmung

Gegeben sei eine Funktion $g\colon \mathbb{R} \to \mathbb{R}$, von der wir eine Nullstelle suchen, d.h. ein $\overline{x} \in \mathbb{R}$ mit $g(\overline{x}) = 0$.

Methode: Newtonverfahren — definiere $\Phi(x) := x - \frac{g(x)}{g'(x)}$.

- Starte mit „beliebigem“ Startwert $x_0 \in \mathbb{R}$.
- Berechne stets $x_{i+1} = \Phi(x_i)$.

Beobachtung:
x_i nähert sich der Nullstelle $\overline{x}$ an

FIXPUNKTITERATION – EINE ANALOGIE

Ausblick: Fixpunktiteration zur Nullstellenbestimmung

Gegeben sei eine Funktion $g\colon \mathbb{R} \to \mathbb{R}$, von der wir eine Nullstelle suchen, d.h. ein $\overline{x} \in \mathbb{R}$ mit $g(\overline{x}) = 0$.

Methode: Newtonverfahren — definiere $\Phi(x) := x - \frac{g(x)}{g'(x)}$.

- Starte mit „beliebigem“ Startwert $x_0 \in \mathbb{R}$.
- Berechne stets $x_{i+1} = \Phi(x_i)$.

Beobachtung:
x_i nähert sich der Nullstelle $\overline{x}$ an

Ein *Fixpunkt* von Φ ist ein Punkt x mit $\Phi(x) = x$.

FIXPUNKTITERATION – EINE ANALOGIE

Ausblick: Fixpunktiteration zur Nullstellenbestimmung

Gegeben sei eine Funktion $g\colon \mathbb{R} \to \mathbb{R}$, von der wir eine Nullstelle suchen, d.h. ein $\overline{x} \in \mathbb{R}$ mit $g(\overline{x}) = 0$.

Methode: Newtonverfahren — definiere $\Phi(x) := x - \frac{g(x)}{g'(x)}$.

- Starte mit „beliebigem" Startwert $x_0 \in \mathbb{R}$.
- Berechne stets $x_{i+1} = \Phi(x_i)$.

Beobachtung:
x_i nähert sich der Nullstelle $\overline{x}$ an

Ein *Fixpunkt* von Φ ist ein Punkt x mit $\Phi(x) = x$.

Die Nullstelle $\overline{x}$ ist ein Fixpunkt von Φ, da

$$\Phi(\overline{x}) = \overline{x} - \frac{g(\overline{x})}{g'(\overline{x})} = \overline{x}.$$

FIXPUNKTITERATION – EINE ANALOGIE

Ausblick: Fixpunktiteration zur Nullstellenbestimmung

Gegeben sei eine Funktion $g\colon \mathbb{R} \to \mathbb{R}$, von der wir eine Nullstelle suchen, d.h. ein $\overline{x} \in \mathbb{R}$ mit $g(\overline{x}) = 0$.

Methode: Newtonverfahren — definiere $\Phi(x) := x - \frac{g(x)}{g'(x)}$.

- Starte mit „beliebigem" Startwert $x_0 \in \mathbb{R}$.
- Berechne stets $x_{i+1} = \Phi(x_i)$.

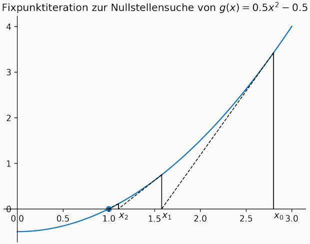
Beobachtung:

x_i nähert sich der Nullstelle $\overline{x}$ an

Ein *Fixpunkt* von Φ ist ein Punkt x mit $\Phi(x) = x$.

Die Nullstelle $\overline{x}$ ist ein Fixpunkt von Φ, da

$$\Phi(\overline{x}) = \overline{x} - \frac{g(\overline{x})}{g'(\overline{x})} = \overline{x}.$$



FIXPUNKTITERATION FÜR EBNF

Ziel: berechne Sprache $W(\mathcal{E}, v)$ für alle $v \in V$ einer EBNF-Definition $\mathcal{E} = (V, \Sigma, S, R)$.

FIXPUNKTITERATION FÜR EBNF

Ziel: berechne Sprache $W(\mathcal{E}, v)$ für alle $v \in V$ einer EBNF-Definition $\mathcal{E} = (V, \Sigma, S, R)$.

Iterierende Funktion:

$$f \colon \underbrace{(V \to \mathcal{P}(\Sigma^*))}_{\rho} \to (V \to \mathcal{P}(\Sigma^*))$$

- Starte mit bisherigen Kenntnis $\rho(v) = \varnothing$ für alle $v \in V$.
  (Nichtswissen)
- Berechne stets neues Wissen $\rho_{\text{neu}} = f(\rho_{\text{alt}})$.
  (Generiere neues Wissen)

FIXPUNKTITERATION FÜR EBNF

Ziel: berechne Sprache $W(\mathcal{E}, v)$ für alle $v \in V$ einer EBNF-Definition $\mathcal{E} = (V, \Sigma, S, R)$.

Iterierende Funktion:

$$f : \underbrace{(V \to \mathcal{P}(\Sigma^*))}_{\rho} \to (V \to \mathcal{P}(\Sigma^*))$$

- Starte mit bisherigen Kenntnis $\rho(v) = \varnothing$ für alle $v \in V$.
  (Nichtswissen)
- Berechne stets neues Wissen $\rho_{\text{neu}} = f(\rho_{\text{alt}})$.
  (Generiere neues Wissen)

Ende: erreiche einen Fixpunkt ρ mit $f(\rho) = \rho$

Dann gilt $\rho(v) = W(\mathcal{E}, v)$ für alle $v \in V$.

FIXPUNKTITERATION FÜR EBNF

Da V endlich ist, ist $f(\rho)\colon V \to \mathcal{P}(\Sigma^*)$ nur auf endlich vielen Argumenten definiert, deren Bilder wir nun als Spaltenvektor schreiben:

$$\begin{pmatrix} f(\rho)(v_1) \\ f(\rho)(v_2) \\ \vdots \\ f(\rho)(v_n) \end{pmatrix} \begin{matrix} \in \mathcal{P}(\Sigma^*) \\ \in \mathcal{P}(\Sigma^*) \\ \vdots \\ \in \mathcal{P}(\Sigma^*) \end{matrix}$$

FIXPUNKTITERATION FÜR EBNF

Da V endlich ist, ist $f(\rho)\colon V \to \mathcal{P}(\Sigma^*)$ nur auf endlich vielen Argumenten definiert, deren Bilder wir nun als Spaltenvektor schreiben:

$$\begin{pmatrix} f(\rho)(v_1) \\ f(\rho)(v_2) \\ \vdots \\ f(\rho)(v_n) \end{pmatrix} \begin{matrix} \in \mathcal{P}(\Sigma^*) \\ \in \mathcal{P}(\Sigma^*) \\ \vdots \\ \in \mathcal{P}(\Sigma^*) \end{matrix}$$

Ein Iterationsprozess lässt sich dann wie folgt notieren:

$$\begin{pmatrix} \varnothing \\ \varnothing \end{pmatrix} \overset{f}{\mapsto}_1 \begin{pmatrix} f(\rho)(v_1) \\ f(\rho)(v_2) \end{pmatrix} \overset{f}{\mapsto}_2 \begin{pmatrix} f(f(\rho))(v_1) \\ f(f(\rho))(v_2) \end{pmatrix} \overset{f}{\mapsto}_3 \dots$$

$$\overset{f}{\mapsto}_n \begin{pmatrix} f^n(\rho)(v_1) \\ f^n(\rho)(v_2) \end{pmatrix} \overset{f}{\mapsto}_{n+1} \dots$$

Pulsierender Speicher

PULSIERENDER SPEICHER

Gültigkeitsbereiche von Objekten:

- Eine Funktion ist ab ihrer Deklaration bis zum Programmende sichtbar. Vorwärtsdeklarationen beachten!
- Ihre formalen Parameter jedoch nur innerhalb der Funktionsdefinition!
- Gibt es gleichlautende formale Parameter in verschiedenen Funktionen, müssen diese in der Tabelle natürlich unterschieden werden (z.B. durch „x in f“).
- Vorsicht bei Namenskonflikten: lokale Variablen überschreiben die Sichtbarkeit globaler Variablen.

PULSIERENDER SPEICHER

Speicherprotokoll:

- Für jeden Funktionsaufruf werden erst die Parameter, dann die lokalen Variablen in Reihenfolge ihres Auftretens in der Umgebung notiert. Globale Variablen stehen ganz vorn.
- Variablennamen werden nur notiert, wenn die Variablen sichtbar sind. Globale Variablennamen werden immer notiert.
- Der Wert von nicht sichtbaren Variablen muss nur notiert werden wenn er sich ändert.
- Uninitialisierte Variablen werden mit Inhalt „?“ notiert.

Prozessproblem

FLOYD-WARSHALL → AHO-HOPCRAFT-ULLMANN

modifizierte Adjazenzmatrix

$$mA_G = \begin{cases} A_G(u,v) & \text{wenn } u \neq v \\ A_G(u,v) \oplus \mathbf{1} & \text{wenn } u = v \end{cases}$$

Initialisierung: $D_G^{(0)} = mA_G$

Rekursion:

$$\boxed{\begin{aligned} & D_G^{(k+1)}(u,v) \\ = \; & D_G^{(k)}(u,v) \oplus \left(D_G^{(k)}(u,k+1) \odot (D_G^{(k)}(k+1,k+1))^* \odot D_G^{(k)}(k+1,v) \right) \end{aligned}}$$

vgl. dazu Floyd-Warshall:

$$\begin{aligned} & D_G^{(k+1)}(u,v) \\ = \; & \min \left\{ D_G^{(k)}(u,v), D_G^{(k)}(u,k+1) + 0 + D_G^{(k)}(k+1,v) \right\} \end{aligned}$$