TECHNISCHE UNIVERSITÄT DRESDEN

ALGORITHMEN UND DATENSTRUKTUREN

ÜBUNG 15: WIEDERHOLUNG & KONSULTATION

Eric Kunze

eric.kunze@tu-dresden.de

TU Dresden, 31.01.2022

letzte Änderung:
30.01.2022, 18:06

VORGESCHLAGENE THEMEN

Fixpunktsemantik

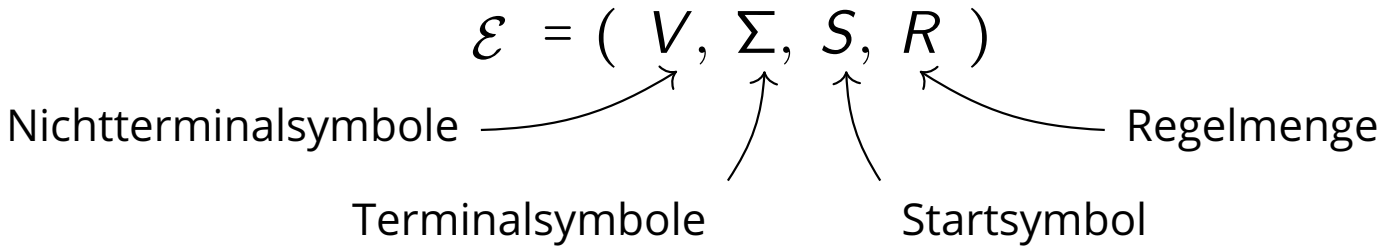
Pulsierender Speicher

C – Programmierung

Prozessproblem

Fixpunktsemantik

EBNF-DEFINITION



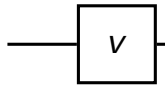
Jede **EBNF-Regel** besteht aus einer linken und einer rechten Seite, die rechte Seite ist ein **EBNF-Term**.

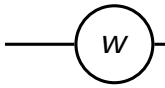
Nichtterminalsymbol ::= *EBNF-Term*

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

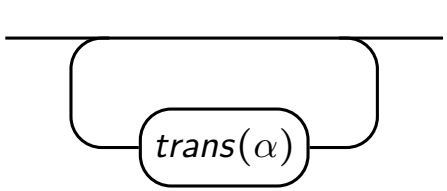
- Wenn $\alpha \in T$, so auch $\hat{(}\ \alpha\ \hat{)} \in T$, $\hat{\{}\ \alpha\ \hat{\}} \in T$, $\hat{[}\ \alpha\ \hat{]} \in T$.
- Wenn $\alpha_1, \alpha_2 \in T$, so auch $\hat{(}\ \alpha_1\ \hat{|}\ \alpha_2\ \hat{)} \in T$, $\alpha_1\alpha_2 \in T$.

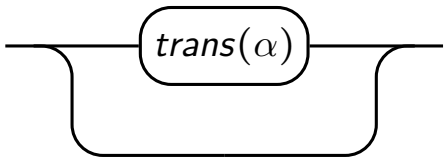
ÜBERSETZUNG EBNF ↔ SYNTAXDIAGRAMME

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

; $trans(w)$ = 

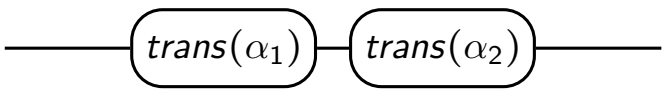
Sei $\alpha \in T(\Sigma, V)$ ein EBNF-Term.

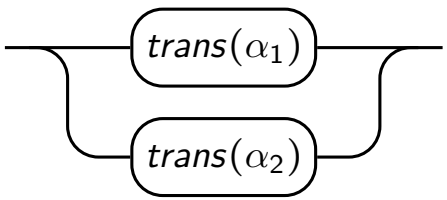
- $trans(\hat{\{}\ \alpha\ \hat{\}})$ = 

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

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

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

- $trans(\ \alpha_1\alpha_2\ )$ = 

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

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

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

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

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 $g(\overline{x}) = 0$.

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

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

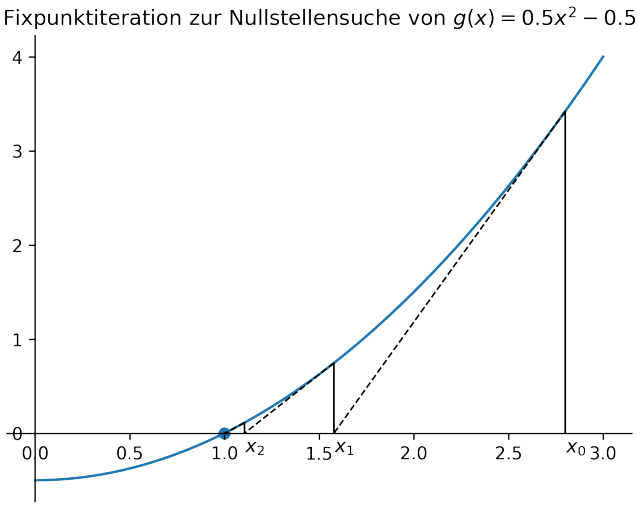
Beobachtung:

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

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:

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

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.

C – Programmierung

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

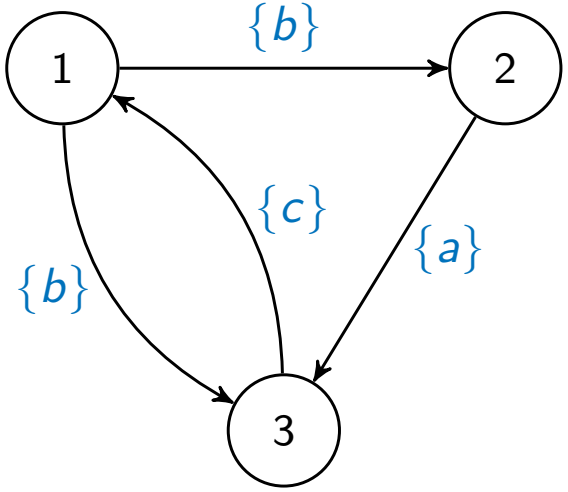
Rekursion:

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

vgl. dazu Floyd-Warshall:

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

AUFGABE AGS 9.5.19



Teil (a) : Geben Sie für G die modifizierte Adjazenzmatrix mA_G an.

$$mA_G = \begin{pmatrix} \{\varepsilon\} & \{b\} & \{b\} \\ \varnothing & \{\varepsilon\} & \{a\} \\ \{c\} & \varnothing & \{\varepsilon\} \end{pmatrix}$$

Teil (b): Berechnen Sie für den Aho-Algorithmus die Matrizen $D_G^{(1)}$ und $D_G^{(2)}$. Sie müssen nur die Matrixelemente aufschreiben, die sich gegenüber mA_G geändert haben.

$$D_G^{(1)} = \begin{pmatrix} \{\varepsilon\} & \{b\} & \{b\} \\ \varnothing & \{\varepsilon\} & \{a\} \\ \{c\} & \{cb\} & \{cb, \varepsilon\} \end{pmatrix} \quad D_G^{(2)} = \begin{pmatrix} \{\varepsilon\} & \{b\} & \{b, ba\} \\ \varnothing & \{\varepsilon\} & \{a\} \\ \{c\} & \{cb\} & \{cb, cba, \varepsilon\} \end{pmatrix}$$

Ergebnis aus Teil (b):

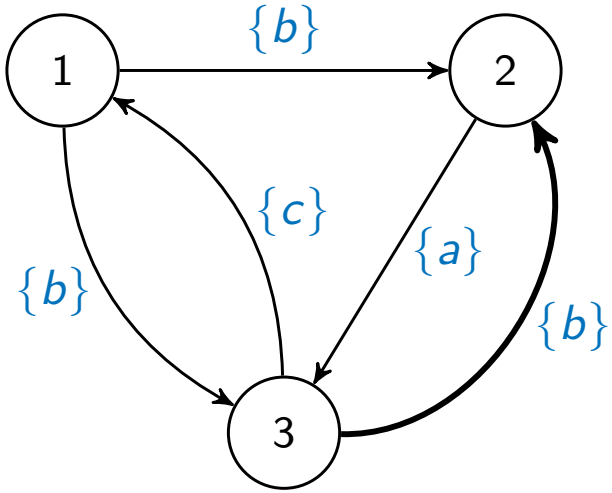
$$D_G^{(1)} = \begin{pmatrix} \{\varepsilon\} & \{b\} & \{b\} \\ \varnothing & \{\varepsilon\} & \{a\} \\ \{c\} & \{cb\} & \{cb, \varepsilon\} \end{pmatrix} \quad D_G^{(2)} = \begin{pmatrix} \{\varepsilon\} & \{b\} & \{b, ba\} \\ \varnothing & \{\varepsilon\} & \{a\} \\ \{c\} & \{cb\} & \{cb, cba, \varepsilon\} \end{pmatrix}$$

Teil (c): Geben Sie die letzte Zeile der Ergebnismatrix D_G des Aho-Algorithmus an.

$$\begin{aligned} D_G(3,1) &= D_G^{(3)}(3,1) \\ &= D_G^{(2)}(3,1) \cup \left\{ D_G^{(2)}(3,3) \cdot (D_G^{(2)}(3,3))^* \cdot D_G^{(2)}(3,1) \right\} \\ &= \{c\} \cup \left\{ \{cb, cba, \varepsilon\} \cdot \{cb, cba, \varepsilon\}^* \cdot \{c\} \right\} \\ &= \{cb, cba\}^* \cdot \{c\} \end{aligned}$$

$$\begin{aligned} D_G(3,2) &= D_G^{(2)}(3,2) \cup \left\{ D_G^{(2)}(3,3) \cdot (D_G^{(2)}(3,3))^* \cdot D_G^{(2)}(3,2) \right\} \\ &= \{cb\} \cup \left\{ \{cb, cba, \varepsilon\} \cdot \{cb, cba, \varepsilon\}^* \cdot \{cb\} \right\} \\ &= \{cb, cba\}^* \cdot \{cb\} \end{aligned}$$

$$\begin{aligned} D_G(3,3) &= D_G^{(2)}(3,3) \cup \left\{ D_G^{(2)}(3,3) \cdot (D_G^{(2)}(3,3))^* \cdot D_G^{(2)}(3,3) \right\} \\ &= \{cb, cba, \varepsilon\} \cup \left\{ \{cb, cba, \varepsilon\} \cdot \{cb, cba, \varepsilon\}^* \cdot \{cb, cba, \varepsilon\} \right\} \\ &= \{cb, cba\}^* \end{aligned}$$



Ergebnis aus Teil (c):

$$D_G(3,1) = \{cb, cba\}^* \cdot \{c\}$$
$$D_G(3,2) = \{cb, cba\}^* \cdot \{cb\}$$
$$D_G(3,3) = \{cb, cba\}^*$$

Teil (d): Wie verändert sich der Eintrag $D_G(3,3)$, wenn zu dem Graphen G eine Kante von Knoten 3 zum Knoten 2 mit dem Gewicht $\{b\}$ zugefügt wird?

$$D_G(3,3) = \{cb, cba, ba\}^* \rightsquigarrow (3, 3, \{cb, cba, ba\}^*)$$