

# ALGORITHMEN UND DATENSTRUKTUREN

## ÜBUNG 15: WIEDERHOLUNG & KONSULTATION

---

Eric Kunze

`eric.kunze@tu-dresden.de`

TU Dresden, 31.01.2022

letzte Änderung:  
30.01.2022, 18:08

Fixpunktsemantik

Pulsierender Speicher

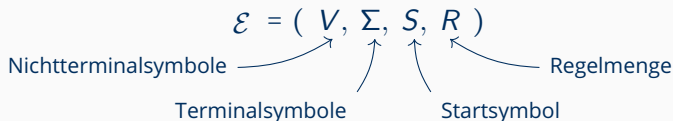
C – Programmierung

Prozessproblem

# Fixpunktsemantik

---

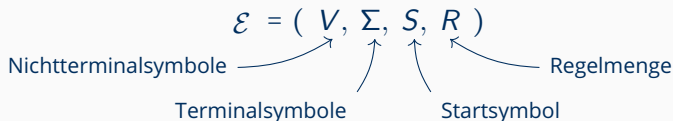
# EBNF-DEFINITION



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

*Nichtterminalsymbol ::= EBNF-Term*

# EBNF-DEFINITION



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

*Nichtterminalsymbol ::= EBNF-Term*

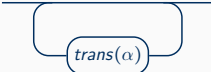
**Definition (EBNF-Terme):** Seien  $V$  (syntaktische Variablen) und  $\Sigma$  (Terminalsymbole) endliche Mengen mit  $V \cap \Sigma = \emptyset$ . Die Menge der EBNF-Terme über  $V$  und  $\Sigma$  (notiere:  $T(\Sigma, V)$ ), ist die *kleinste* Menge  $T \subseteq \left( V \cup \Sigma \cup \left\{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

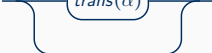
- ▶ Wenn  $\alpha \in T$ , so auch  $\hat{(\alpha)} \in T, \hat{\{\alpha\}} \in T, \hat{[\alpha]} \in T$ .
- ▶ Wenn  $\alpha_1, \alpha_2 \in T$ , so auch  $\hat{(\alpha_1 \mid \alpha_2)} \in T, \alpha_1 \alpha_2 \in T$ .

# ÜBERSETZUNG EBNF $\leftrightarrow$ SYNTAXDIAGRAMME

Sei  $v \in V$  und  $w \in \Sigma$ .  $trans(v) = \text{---} \boxed{v} \text{---}$ ;  $trans(w) = \text{---} \bigcirc w \text{---}$

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


►  $trans(\hat{\{ \alpha \}}) =$  The diagram shows a horizontal line entering from the left, which then splits into two parallel paths. The upper path contains a rounded rectangle labeled  $trans(\alpha)$ . The lower path is a direct connection. The two paths then merge back into a single horizontal line exiting to the right.

►  $trans(\hat{[ \alpha ]}) =$  The diagram shows a horizontal line entering from the left, which then splits into two parallel paths. The upper path contains a rounded rectangle labeled  $trans(\alpha)$ . The lower path is a direct connection. The two paths then merge back into a single horizontal line exiting to the right.

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

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

►  $trans(\alpha_1 \alpha_2) =$  The diagram shows a horizontal line entering from the left, passing through a rounded rectangle labeled  $trans(\alpha_1)$ , then through another rounded rectangle labeled  $trans(\alpha_2)$ , and finally exiting to the right.

►  $trans(\hat{( \alpha_1 \mid \alpha_2 )}) =$  The diagram shows a horizontal line entering from the left, which then splits into two parallel paths. The upper path contains a rounded rectangle labeled  $trans(\alpha_1)$ . The lower path contains a rounded rectangle labeled  $trans(\alpha_2)$ . The two paths then merge back into a single horizontal line exiting to the right.

**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
- ▶  $\rho: V \rightarrow \mathcal{P}(\Sigma^*)$  ordnet jeder syntaktischen Variable  $v \in V$  eine Sprache zu
- ▶ Vorstellung:  $\rho(v)$  ist bestes Wissen über die von  $v$  beschriebene Sprache

**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
- ▶  $\rho: V \rightarrow \mathcal{P}(\Sigma^*)$  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  $\llbracket \cdot \rrbracket: \underbrace{T(\Sigma, V)}_{\text{EBNF-Term } \alpha} \rightarrow \underbrace{((V \rightarrow \mathcal{P}(\Sigma^*)) \rightarrow \mathcal{P}(\Sigma^*))}_{\rho}$



$$\llbracket \cdot \rrbracket : \underbrace{T(\Sigma, V)}_{\text{EBNF-Term } \alpha} \rightarrow \underbrace{((V \rightarrow \mathcal{P}(\Sigma^*)) \rightarrow \mathcal{P}(\Sigma^*))}_{\rho}$$

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

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

# FIXPUNKTITERATION – EINE ANALOGIE

**Ausblick:** Fixpunktiteration zur Nullstellenbestimmung

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

# FIXPUNKTITERATION – EINE ANALOGIE

**Ausblick:** Fixpunktiteration zur Nullstellenbestimmung

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

**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  $g: \mathbb{R} \rightarrow \mathbb{R}$ , von der wir eine Nullstelle suchen, d.h. ein  $\bar{x} \in \mathbb{R}$  mit  $g(\bar{x}) = 0$ .

**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  $\bar{x}$  an

# FIXPUNKTITERATION – EINE ANALOGIE

**Ausblick:** Fixpunktiteration zur Nullstellenbestimmung

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

**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  $\bar{x}$  an

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

# FIXPUNKTITERATION – EINE ANALOGIE

**Ausblick:** Fixpunktiteration zur Nullstellenbestimmung

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

**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  $\bar{x}$  an

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

Die Nullstelle  $\bar{x}$  ist ein Fixpunkt von  $\Phi$ , da

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

# FIXPUNKTITERATION – EINE ANALOGIE

**Ausblick:** Fixpunktiteration zur Nullstellenbestimmung

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

**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:**

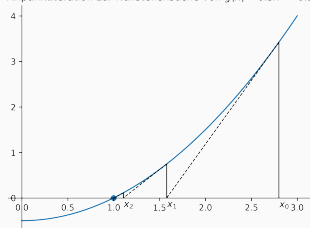
$x_i$  nähert sich der Nullstelle  $\bar{x}$  an

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

Die Nullstelle  $\bar{x}$  ist ein Fixpunkt von  $\Phi$ , da

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

Fixpunktiteration zur Nullstellensuche von  $g(x) = 0.5x^2 - 0.5$



# 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:

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

- ▶ Starte mit bisherigen Kenntnis  $\rho(v) = \emptyset$  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:

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

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

**Ende:** erreiche einen Fixpunkt  $\rho$  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  $f(\rho): V \rightarrow \mathcal{P}(\Sigma^*)$  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  $f(\rho): V \rightarrow \mathcal{P}(\Sigma^*)$  nur auf endlich vielen Argumenten definiert, deren Bilder wir nun als Spaltenvektor schreiben:

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

Ein Iterationsprozess lässt sich dann wie folgt notieren:

$$\begin{pmatrix} \emptyset \\ \emptyset \end{pmatrix} \xrightarrow{f_1} \begin{pmatrix} f(\rho)(v_1) \\ f(\rho)(v_2) \end{pmatrix} \xrightarrow{f_2} \begin{pmatrix} f(f(\rho))(v_1) \\ f(f(\rho))(v_2) \end{pmatrix} \xrightarrow{f_3} \dots$$
$$\xrightarrow{f_n} \begin{pmatrix} f^n(\rho)(v_1) \\ f^n(\rho)(v_2) \end{pmatrix} \xrightarrow{f_{n+1}} \dots$$

# 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.

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

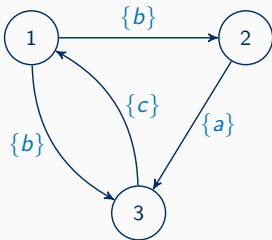
**Rekursion:**

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

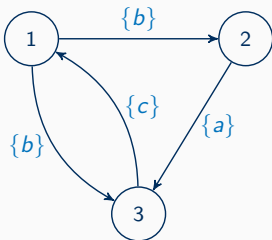
vgl. dazu Floyd-Warshall:

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

## AUFGABE AGS 9.5.19

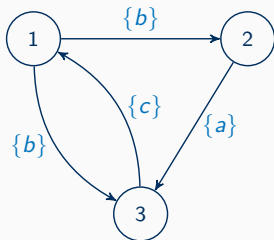


## AUFGABE AGS 9.5.19



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

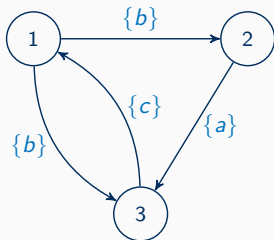
## AUFGABE AGS 9.5.19



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

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

## AUFGABE AGS 9.5.19

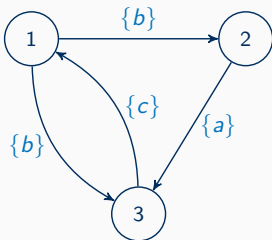


**Teil (a)** : Geben Sie für  $G$  die modifizierte Adjazenzmatrix  $mA_G$  an.

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

**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.

## AUFGABE AGS 9.5.19



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

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

**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.

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

Ergebnis aus Teil (b):

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

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



Ergebnis aus Teil (b):

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

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

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

Ergebnis aus Teil (b):

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

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

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

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

Ergebnis aus Teil (b):

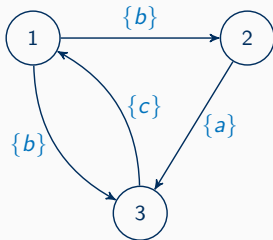
$$D_G^{(1)} = \begin{pmatrix} \{\varepsilon\} & \{b\} & \{b\} \\ \emptyset & \{\varepsilon\} & \{a\} \\ \{c\} & \{cb\} & \{cb, \varepsilon\} \end{pmatrix} \quad D_G^{(2)} = \begin{pmatrix} \{\varepsilon\} & \{b\} & \{b, ba\} \\ \emptyset & \{\varepsilon\} & \{a\} \\ \{c\} & \{cb\} & \{cb, cba, \varepsilon\}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

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

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

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

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

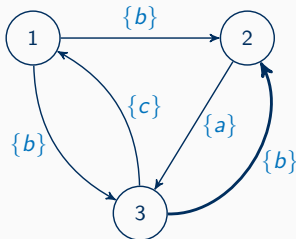


Ergebnis aus Teil (c):

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

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

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



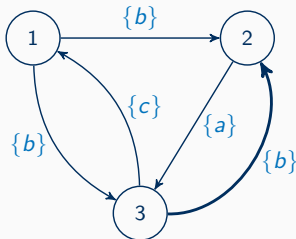
Ergebnis aus Teil (c):

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

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

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

**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?



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?

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