

ALGORITHMEN UND DATENSTRUKTUREN

ÜBUNG 2: SYNTAXDIAGRAMME

Eric Kunze

`eric.kunze@mailbox.tu-dresden.de`

TU Dresden, 06.11.2019

Syntaxdiagramme

- ▶ syntaktische Variable = Nichtterminalsymbol = Name eines Syntaxdiagramms
- ▶ Jedes Kästchen ist mit dem Namen eines Syntaxdiagramms beschriftet.
- ▶ Jedes Oval ist mit einem Terminalsymbol beschriftet.

Rücksprunugalgorithmus

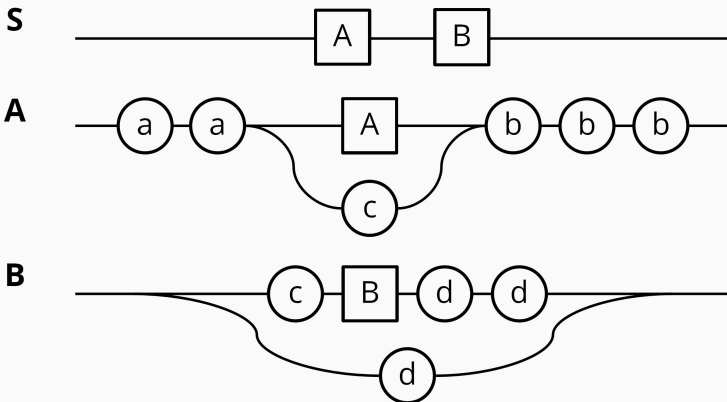
- ▶ jedes Kästchen bekommt eindeutige Marke (Rücksprungadresse)
- ▶ beim Betreten eines Syntaxdiagramms wird eine Marke auf den Keller gelegt
- ▶ Nachweis von Zugehörigkeit eines Wortes zu einer Sprache

AUFGABE 1

- ▶ Teil (a) — z.B. $\varepsilon, a, c, caa, aaaa, \dots$
- ▶ Teil (b) — z.B. $aaac, abacac, abbaccac, \dots$
- ▶ Teil (c) — z.B. $\varepsilon, ab, abab, ac, aabcab, \dots$

AUFGABE 2 — TEIL (A)

$$\begin{aligned} L &= \{a^{2i}cb^{3i}c^kd^{2k+1} \mid i > 0, k \geq 0\} \\ &= \{a^{2i}cb^{3i} \mid i > 0\} \cdot \{c^kd^{2k+1} \mid k \geq 0\} \end{aligned}$$



AUFGABE 2 — TEIL (B)

Protokollierungszeitpunkte:

- ▶ jeder Aufenthalt in einem Syntaxdiagramm entspricht einer Zeile
- ▶ jede Zeile führt eine Operation auf dem Markenkeller aus
- ▶ \mathcal{Z} = Rücksprung zu Marke 3

Wort	Markenkeller
a	1
a	31
aa	131
aaa	2131
aaa	32131
aaaaccb	\mathcal{Z} 2131
aaaaccb	\mathcal{Z} 131
aaaaccbd	\mathcal{Z} 131
aaaaccbdb	\mathcal{Z} 1
aaaaccbdb	\mathcal{Z}
aaaaccbdbb	–

Alphabet der Aussagenlogik

Ein Alphabet der Aussagenlogik besteht aus

- ▶ einer (abzählbar) unendlichen Menge $\mathcal{R} = \{p_1, p_2, p_3, \dots\}$ von aussagenlogischen Variablen
- ▶ der Menge $\mathcal{J} = \{\neg, \wedge, \vee, \rightarrow, \leftrightarrow\}$ von Junktoren
- ▶ der Menge $\{(,)\}$ der Sonderzeichen.

Wir beschränken uns im Folgenden auf die Junktoren $\mathcal{J} = \{\neg, \vee\}$ und die Variablen p und q .

Aussagenlogische Formeln

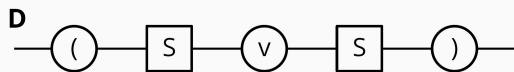
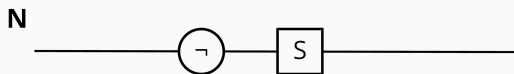
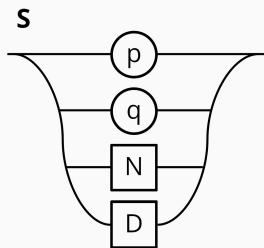
Die Menge von Formeln ist die *kleinste* Menge $\mathcal{L}(\mathcal{R})$ von Zeichenreihen über \mathcal{R} , den Junktoren und den Sonderzeichen, die die folgenden Eigenschaften erfüllt:

- ▶ Wenn $F \in \mathcal{R}$, dann ist $F \in \mathcal{L}(\mathcal{R})$
- ▶ Wenn $F \in \mathcal{L}(\mathcal{R})$, dann ist $\neg F \in \mathcal{L}(\mathcal{R})$
- ▶ Wenn $\circ \in \mathcal{J}$ ein zweistelliger Junktor ist und $F, G \in \mathcal{L}(\mathcal{R})$ sind, dann ist $(F \circ G) \in \mathcal{L}(\mathcal{R})$.

Da im Folgenden stets $\mathcal{J} = \{\neg, \vee\}$ gilt, und \vee der einzige zweistellige Junktor ist, vereinfacht sich die dritte Bedingung zu:

- ▶ Wenn $F, G \in \mathcal{L}(\mathcal{R})$ sind, dann ist $(F \vee G) \in \mathcal{L}(\mathcal{R})$.

AUFGABE 3



Extended Backus-Naur-Form

EBNF-DEFINITION

- ▶ EBNF-Definition besteht aus endlicher Menge von EBNF-Regeln.
- ▶ Jede EBNF-Regel besteht aus einer linken und einer rechten Seite, die rechte Seite ist ein EBNF-Term.

Definition: EBNF-Term

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

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

AUFGABE 4

Sei $V = \{A, B\}$ und $\Sigma = \{a, b, c, d\}$.

- a. $\{ A \} \in T(\Sigma, V)$
- b. $\{ [B] \} \in T(\Sigma, V)$
- c. $\{ ([B] \mid C) \} \notin T(\Sigma, V)$, da $C \notin V$
- d. $\{ (a \mid B \cup \{c\}) \} \notin T(\Sigma, V)$, da \cup nicht in EBNF vorhanden
- e. $\{ ([c] \mid (a \mid b) a) \} \in T(\Sigma, V)$
- f. $c \{ [A \mid B] \}^{\sim} d \notin T(\Sigma, V)$, da \sim und $\{ \}$ zu \mid fehlen
- g. $\{ (a \mid b)^* \mid ABA \} \notin T(\Sigma, V)$, da $*$ nicht in EBNF vorhanden