Blatt 8

Aufgabe 4

$w = babaa$

Tabelle [1,5] enthält S
→ w ableitbar

	1	2	3	4	5
1	b {A}	ba {S,C}	bab {S,C}	baba {A}	babaa {S,C}
2		a {B,C}	ab {B}	aba {S,A}	abaa {B,S,C}
3			b {A}	ba {S,C}	baa {A}
4				a {B,C}	aa {S,A}
5					a {B,C}

Nebenrechnungen: (Durchprobieren von Split-Positionen)

bab: b|ab ⇝ S,C
ba|b ⇝ ∅

baa: b|aa ⇝ ∅
ba|a ⇝ A

aba: a|ba ⇝ S,A
ab|a ⇝ S

baba: b|aba ⇝ ∅
ba|ba ⇝ A
bab|a ⇝ A

abaa: a|baa ⇝ B
ab|aa ⇝ B
aba|a ⇝ S,C

babaa: b|abaa ⇝ S,C
ba|baa ⇝ ∅
bab|aa ⇝ ∅
baba|a ⇝ S,C

ω = baba

Tabelle [1,4] enthält kein S
→ ω nicht ableitbar

	1	2	3	4
1	b {A}	ba {S,C}	bab {S,C}	{A}
2		a {B,C}	ab {B}	aba {S,A}
3			b {A}	ba {S,C}
4				a {B,C}

bab : b|ab ⇝ S,C
ba|b ⇝ ∅

aba : a|ba ⇝ S,A
ab|a ⇝ S

baba : b|aba ⇝ ∅
ba|ba ⇝ A
bab|a ⇝ A

Aufgabe 2

Idee von CYK:

1) In Tabelle $[i,j]$ $(i \leq j)$ sollen alle Nicht-Terminale A stehen für die gilt $A \rightarrow^* a_i \dots a_j$

2) dafür teilt der Algorithmus $a_i \dots a_j$ an jeder möglichen Position (beide Teilworte $a_i \dots a_k$ und $a_{k+1} \dots a_j$ dürfen nicht leer sein) $(i \leq k < j)$
und liest in Tabelle $[i,k]$ und Tabelle $[k+1,j]$ nach welche Nicht-Terminale $a_i \dots a_k$ resp. $a_{k+1} \dots a_j$ erzeugen (z.B $B \rightarrow a_i \dots a_k$ und $C \rightarrow a_{k+1} \dots a_j$)

3) falls $A \rightarrow BC$ eine Grammatikregel ist, wird A zu Tabelle $[i,j]$ hinzugefügt

Mit anderen Worten, der CYK-Algorithmus versucht von dem Wort w ausgehend "rückwärts" Ableitungen zu finden.
Wenn S in Tabelle $[1, |w|]$ enthalten ist, wurde eine Ableitung $S \rightarrow^* w$ gefunden.

a) Idee vom "erweiterten" Algorithmus.
Wir wollen die (oder eine) vom CYK-Algorithmus gefundene Ableitung aus der Tabelle ablesen.
Problem:
Weder die Regel $A \rightarrow BC$ aus Punkt 3) oben noch die Splitting-Position aus Punkt 2) wurden gespeichert
→ sind aber nötig um die Ableitung zu rekonstruieren.

Lösung: erweitere den CYK-Algorithmus so dass die angewandte Regel und die splitting Position gespeichert werden.

→ Tabelle [i,j] enthält
- Liste von Nicht-Terminalen (wie gehabt)
- Liste von Tupeln mit $(A \to BC, k)$
  - $A \to BC$: angewandte Regel
  - k: splitting Position

Im CYK-Algorithmus

- Tabelle [i,i] wird initialisiert durch $(A \to a_i, i)$ für jede Grammatik-Regel mit rechter Seite a_i.
- wenn in Tabelle [i,j] Nicht-Terminal A hinzugefügt wird

  weil
  - $A \to BC \in P$
  - Tabelle [i,k] enthält B
  - Tabelle [k+1, j] enthält C

  (für $i \leq k < j$)

  dann: füge zusätzlich $(A \to BC, k)$ in die zweite Liste von Tabelle[i,j] ein.

Der Algorithmus der die benutzten Regeln zu einer Links-Ableitung für $w \in L(G)$ berechnet sieht wie folgt aus:

- fülle Tabelle mit erweitertem CYK-Algo
- führe Regeln $(S, (1,n))$ aus $(n = |w|)$

Regeln $(A, (i,j))$ // i≤j Indexe in der Tabelle, $A \in N$

- suche ein (beliebiges) Tupel $(A \to \alpha, k)$ aus der Liste von Tabelle[i,j] aus // existiert da $w \in L(G)$
- falls $\alpha \in \Sigma$ return $A \to \alpha$
- sonst ist $\alpha = BC$ // da CNF

  return $A \to BC$, Regeln (i,k), Regeln $(k+1,j)$

b) • Wir benutzen wieder dass der CYK-Algorithmus alle Ableitungen $S \to^* w$ "rückwärts" berechnet

→ verwende wieder erweiterten CYK-Algorithmus

• Überlegung:

Wenn $A \to BC$ einzige Regel mit linker Seite A

und von B aus gibt es n_1 Links-Ableitungen zu $w_1 \in \Sigma^*$

von C ——— n_2 ——— $w_2 \in \Sigma^*$

dann gibt es $n_1 \cdot n_2$ Links-Ableitungen von A zu $w_1 . w_2$

Falls es mehrere Ableitungen mit linker Seite A gibt, müssen wir die Summe über die Ableitungen bilden.

Der Algorithmus der die Anzahl der Linksableitungen von S zu einem Wort w berechnet sieht wie folgt aus:

• fülle Tabelle mit erweitertem CYK-Algorithmus
• führe Anzahl $(S, (1, n))$ $(n = |w|)$ aus

Anzahl $(A, (i,j))$

$m := 0$ // initialisiere Anzahl an Links-Ableitungen

• gehe über alle Tupel $(A \to \alpha, k)$ mit linker Seite A
  - wenn $\alpha \in \Sigma$: $m := m + 1$
  - wenn $\alpha = BC$: $m := m +$ Anzahl $(B, (i,k)) \cdot$ Anzahl $(C, (k+1, j))$

• return m

c)

Tabelle vom erweiterten CYK-Algorithmus auf Wort $w = aaa$:

	1	2	3
1	A $(A \to a, 1)$	S, A $(S \to AA, 1)$ $(A \to AA, 1)$	S, A $(S \to AA, 1), (A \to AA, 1)$ $(S \to SA, 2)$ $(S \to AA, 2), (A \to AA, 2)$
2		A $(A \to a, 2)$	S, A $(S \to AA, 2)$ $(A \to AA, 2)$
3			A $(A \to a, 3)$

(b) Anzahl der Linksableitungen von S zu aaa:

$$\text{Anzahl}(S, (1,3)) = \text{Anzahl}(A, (1,1)) \cdot \text{Anzahl}(A, (2,3)) + \text{Anzahl}(S, (1,2)) \cdot \text{Anzahl}(A, (3,3)) + \text{Anzahl}(A, (1,2)) \cdot \text{Anzahl}(A, (3,3))$$

$$= 1 \cdot (\text{Anzahl}(A, (2,2)) \cdot \text{Anzahl}(A, (3,3)) + (\text{Anzahl}(A, (1,1)) \cdot \text{Anzahl}(A, (2,2))) \cdot 1 + (\text{Anzahl}(A, (1,1)) \cdot \text{Anzahl}(A, (2,2))) \cdot 1$$

$$= 1 \cdot (1 \cdot 1) + (1 \cdot 1) \cdot 1 + (1 \cdot 1) \cdot 1 = 3 \quad (*)$$

(a) eine mögliche Abfolge von Regeln zu einer Links-Ableitung von $S \to aaa$:

$\ldots \text{Regeln}(S, (1,3)) = S \to SA, \ldots \text{Regeln}(S, (1,2)), \ldots \text{Regeln}(A, (3,3))$

$= S \to SA, S \to AA, \text{Regeln}(A, (1,1)), \ldots \text{Regeln}(A, (2,2)), A \to a$

$= S \to SA, S \to AA, A \to a, A \to a, A \to a$

zugehörige Links-Ableitung

$S \to SA \to AAA \to aAA \to aaA \to aaa$

$(*)$ Tatsächlich:

$S \to SA \to AAA \to aAA \to aaA \to aaa$

$S \to AA \to aA \to aAA \to aaA \to aaa$

$S \to AA \to AAA \to aAA \to aaA \to aaa$

sind alle Linksableitungen von S zu aaa