

# Grundlagen der Theoretischen Informatik

## 4. Kellerautomaten und kontextfreie Sprachen (VII)

22.06.2017

Viorica Sofronie-Stokkermans

e-mail: [sofronie@uni-koblenz.de](mailto:sofronie@uni-koblenz.de)

# Übersicht

---

1. Motivation
2. Terminologie
3. Endliche Automaten und reguläre Sprachen
4. Kellerautomaten und kontextfreie Sprachen
5. Turingmaschinen und rekursiv aufzählbare Sprachen
6. Berechenbarkeit, (Un-)Entscheidbarkeit
7. Komplexitätsklassen P und NP

# Wortprobleme

---

## Problem

**Gegeben:** eine cf-Grammatik  $G$ , so dass  $L(G)$  eine Sprache ist über  $\Sigma$ , und ein Wort  $w \in \Sigma^*$

**Frage:** Ist  $w \in L(G)$ ?

# Wortproblem

---

## Das Wortproblem für $\mathcal{L}_2$

- Zu jeder cf-Grammatik  $G$  kann man einen PDA konstruieren
- Aber ein Pushdown-Automat kann  $\varepsilon$ -Übergänge machen, in denen er das Wort nicht weiter liest.
- **Wie kann man dann garantieren, dass der Automat in endlich vielen Schritten das Wort  $w$  zu Ende gelesen hat?**
- Deshalb: verwende anderes Verfahren:  
**Cocke-Younger-Kasami-Algorithmus** (CYK-Algorithmus)  
Auch: Chart-Parsing

# Chart-Parsing

---

Gegeben: Ein Wort

$$w = a_1 \dots a_n$$

## Idee

- Prinzip der dynamischen Programmierung
- 1.: Ermittle woraus sich die einstelligen Teilworte ableiten lassen
- 2.: Ermittle woraus sich die zweistelligen Teilworte ableiten lassen
- ...
- $n$ .: Ermittle woraus sich die  $n$ -stelligen Teilworte ( $w$  selbst) ableiten lassen

# Chart-Parsing

---

## Zur Vereinfachung

Wir fordern:            Grammatik ist in **Chomsky-Normalform**

Dann:

Immer nur **zwei** benachbarte Kanten betrachten, um herauszufinden, ob darüber eine neue Kante eingefügt werden kann.

# Chart-Parsing

---

**Beispiel (Forts.)**  $G = (\{S, S_a, S_b, A, B\}, \{a, b\}, R, S)$  mit

$$R = \{ \begin{array}{l} S \rightarrow AS_a \mid BS_b \mid AA \mid BB \\ S_a \rightarrow SA \\ S_b \rightarrow SB \\ A \rightarrow a \\ B \rightarrow b \end{array} \}$$

# Chart-Parsing

---

## Darstellung als Array

Für eine Kante, die den  $i$ . bis  $j$ . Buchstaben überspannt und mit  $A$  markiert ist, steht im  $[i, j]$ -Element des Arrays die Eintragung  $A$ .

### **Definition ( $M * N$ )**

Sei  $L = L(G)$  kontextfrei, und  $G = (V, T, R, S)$  in Chomsky-Normalform.

Mit  $M, N \subseteq V$  sei

$$M * N := \{A \in V \mid \exists B \in M, \exists C \in N : A \rightarrow BC \in R\}$$



# Chart-Parsing

---

## Definition ( $w_{i,j}$ , $V_{i,j}$ )

Sei  $w = a_1 \dots a_n$  mit  $a_i \in \Sigma$ .

Dann:

- $w_{i,j} := a_i \dots a_j$  ist das Fragment von  $w$  vom  $i$ -ten bis zum  $j$ -ten Buchstaben
- $V_{i,j} := \{A \in V \mid A \xRightarrow*_G w_{i,j}\}$

# Chart-Parsing

---

Sei  $w = a_1 \dots a_n$ ,  $a_i \in \Sigma$ , d.h.  $|w| = n$ . Dann gilt:

1.  $V_{i,i} = \{A \in V \mid A \rightarrow a_i \in R\}$

2.  $V_{i,k} = \bigcup_{j=i}^{k-1} V_{i,j} * V_{j+1,k}$  für  $1 \leq i < k \leq n$

## Beachte:

Die Grammatik muss in Chomsky-Normalform sein!

# Chart-Parsing

---

Beweis.

1.  $V_{i,i} = \{A \in V \mid A \Longrightarrow_G^* a_i\} = \{A \in V \mid A \rightarrow a_i \in R\}$ , da  $G$  in CNF ist.

$A \in V_{i,k}$  mit  $1 \leq i < k \leq n$

gdw  $A \Longrightarrow_G^* a_i \dots a_k$

gdw  $\exists j, i \leq j < k : \exists B, C \in V : A \Longrightarrow BC$ , und

$B \Longrightarrow_G^* w_{i,j} \neq \varepsilon$

und  $C \Longrightarrow_G^* w_{j+1,k} \neq \varepsilon$  (da  $G$  in CNF ist)

gdw  $\exists j, i \leq j < k : \exists B, C \in V : A \Longrightarrow BC$

und  $B \in V_{i,j}$  und  $C \in V_{j+1,k}$

2. gdw  $\exists j, i \leq j < k : A \in V_{i,j} * V_{j+1,k}$

# CYK-Algorithmus (Cocke-Younger-Kasami)

---

## Algorithmus

Input sei eine Grammatik  $G$  in CNF und ein Wort  $w = a_1 \dots a_n \in \Sigma^*$ .

(i) **for**  $i := 1$  **to**  $n$  **do**                    / \* Regeln  $A \rightarrow a$  eintragen \* /

$$V_{i,i} := \{A \in V \mid A \rightarrow a_i \in R\}$$

(ii) **for**  $h := 1$  **to**  $n - 1$  **do**

**for**  $i := 1$  **to**  $n - h$  **do**

$$V_{i,i+h} = \bigcup_{j=i}^{i+h-1} V_{i,j} * V_{j+1,i+h}$$

(iii) **if**  $S \in V_{1,n}$  **then return** Ausgabe  $w \in L(G)$

**else return** Ausgabe  $w \notin L(G)$

# CYK-Algorithmus (Cocke-Younger-Kasami)

---

Beispiel:

$G = (\{S, A, B\}, \{a, b\}, R, S)$  mit

$R : S \rightarrow AB \mid BC$

$A \rightarrow BA \mid a$

$B \rightarrow CC \mid b$

$C \rightarrow AB \mid a$

An der Tafel.

# CYK-Algorithmus (Cocke-Younger-Kasami)

---

## Eigenschaften

Für Wörter der Länge  $|w| = n$  entscheidet der CYK-Algorithmus in der Größenordnung von  $n^3$  Schritten, ob  $w \in L(G)$  ist.

# CYK-Algorithmus (Cocke-Younger-Kasami)

---

## Beispiel.

Grammatik in CNF  $G = (\{S, S_a, S_b, A, B\}, \{a, b\}, R, S)$

$$R = \{ \begin{array}{l} S \rightarrow AS_a \mid BS_b \mid AA \mid BB \\ S_a \rightarrow SA \\ S_b \rightarrow SB \\ A \rightarrow a \\ B \rightarrow b \end{array} \}$$

$$L(G) = \{vv^R \mid v \in \{a, b\}^+\}$$

Beispiel an der Tafel.

# Übersicht

---

1. Motivation
2. Terminologie
3. Endliche Automaten und reguläre Sprachen
4. Kellerautomaten und kontextfreie Sprachen
5. Turingmaschinen und rekursiv aufzählbare Sprachen
6. Berechenbarkeit, (Un-)Entscheidbarkeit
7. Komplexitätsklassen P und NP



# Übersicht

---

1. Motivation
2. Terminologie
3. Endliche Automaten und reguläre Sprachen
4. Kellerautomaten und kontextfreie Sprachen
5. Turingmaschinen und rekursiv aufzählbare Sprachen
6. Berechenbarkeit, (Un-)Entscheidbarkeit
7. Komplexitätsklassen P und NP