

# Automatentheorie & Formale Sprachen

WiSe 2012/13, Wiebke Petersen

Lehrbuch: Rolf Socher, Theoretische Grundlagen der Informatik, Hanser, 3. Auflage, 2008

## 2. Kapitel

---

**Definition 2.1:** Deterministischer endlicher Automat, DEA

---

Ein *deterministischer endlicher Automat* besteht aus den fünf Komponenten  $Z, \Sigma, \delta, z_0, E$ .

- $Z$  ist eine endliche Menge von *Zuständen*.
  - $\Sigma$  ist ein Alphabet, das sog. *Eingabealphabet*.
  - $\delta : Z \times \Sigma \rightarrow Z$  ist die *Übergangsfunktion*.
  - $z_0 \in Z$  ist der *Startzustand*.
  - $E \subseteq Z$  ist die Menge der *Endezustände*.
- 

(Socher, S. 19)

**Anmerkungen & Fragen:**

- Kann die Zustandsmenge  $Z$  leer sein?
- Kann das Eingabealphabet leer sein?

---

**Definition 2.2:** Konfiguration

---

Eine *Konfiguration* eines endlichen Automaten  $A$  ist ein Paar  $(z, w)$  mit  $z \in Z, w \in \Sigma^*$ . Dabei bezeichnet  $z$  den aktuellen Zustand, in dem sich der Automat zu einem Zeitpunkt der Berechnung befindet, und  $w$  das verbleibende Eingabewort.

Die *Übergangsrelation*  $\rightarrow$  auf der Menge der Konfigurationen von  $A$  ist definiert durch

$$(z, aw) \rightarrow (z', w), \text{ falls } z \rightarrow_a z'.$$

---

(Socher, S. 20)

**Anmerkungen & Fragen:**

- Definieren sie die transitive, reflexive Hülle der Übergangsfunktion  $\rightarrow$

---

**Definition 2.3:** Akzeptierung durch einen DEA, reguläre Sprache

---

Der DEA  $A$  *akzeptiert* das Eingabewort  $w$  genau dann, wenn es ein  $z_e \in E$  gibt mit

$$(z_0, w) \rightarrow^* (z_e, \varepsilon).$$

Die vom Automaten  $A$  *akzeptierte Sprache*  $\mathcal{L}(A)$  ist definiert durch:

$$\mathcal{L}(A) = \{w \in \Sigma^* \mid A \text{ akzeptiert } w\}.$$

Eine Sprache  $L$  heißt *regulär*, falls es einen DEA  $A$  gibt, der  $L$  akzeptiert, das heißt, wenn  $L = \mathcal{L}(A)$  gilt.

---

(Socher, S. 21)

---

**Definition 2.4:** Äquivalenz von Automaten

---

Zwei Automaten  $A_1$  und  $A_2$  heißen *äquivalent*, geschrieben  $A_1 \equiv A_2$ , wenn sie dieselbe Sprache akzeptieren, das heißt, wenn  $\mathcal{L}(A_1) = \mathcal{L}(A_2)$  gilt.

Die Relation  $\equiv$  ist eine Äquivalenzrelation (siehe Abschnitt 8.3).

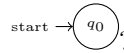
---

(Socher, S. 22)

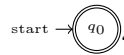
### Anmerkungen & Fragen:

- In welchem Verhältnis steht die Menge der Äquivalenzklassen zur Menge der regulären Sprachen?
- Wie mächtig ist die Äquivalenzklasse des endlichen Automaten zur Paritätsprüfung (Abbildung 2.2)?

- Wie mächtig ist die Äquivalenzklasse des endlichen Automaten



- Wie mächtig ist die Äquivalenzklasse des endlichen Automaten



---

### Definition 2.5: Erreichbarkeit, Fehlerzustand

Der Zustand  $z'$  heißt *erreichbar* vom Zustand  $z$ , falls es ein Wort  $w$  gibt, so dass  $(z, w) \rightarrow^* (z', \epsilon)$  gilt.  $[z]^*$  bezeichnet die Menge aller von  $z$  erreichbaren Zustände.

Ein *Fehlerzustand* eines Automaten  $A$  ist ein Zustand, von dem aus kein Endzustand erreichbar ist, also ein Zustand  $z$ , für den  $[z]^* \cap E = \emptyset$  gilt.

(Socher, S. 22)

### Anmerkungen & Fragen:

- Gilt für jeden DEA  $Z = [z_0]^*$ ?
- Gilt für jeden DEA  $Z \supseteq [z_0]^*$ ?
- Gilt für jeden DEA  $E \subseteq [z_0]^*$ ?

---

### Definition 2.6: Akzeptierung im Zustand $z$

Der DEA  $A$  *akzeptiert* das Eingabewort  $w$  im Zustand  $z$  wenn es ein  $z_e \in E$  gibt mit  $(z, w) \rightarrow^* (z_e, \epsilon)$ .

Die von  $A$  im Zustand  $z$  *akzeptierte Sprache*  $\mathcal{L}(A, z)$  ist definiert durch:

$$\mathcal{L}(A, z) = \{w \in \Sigma^* \mid A \text{ akzeptiert } w \text{ im Zustand } z\}.$$

(Socher, S. 22f)

### Anmerkungen & Fragen:

- Drücken sie die Akzeptanz des Worts  $w$  im Zustand  $z$  mithilfe der erweiterten Übergangsfunktion  $\delta$  aus

---

### Definition 2.7: Minimalautomat

Sei  $L$  eine reguläre Sprache über dem Alphabet  $\Sigma$  und sei  $A$  ein Automat, der  $L$  akzeptiert, das heißt  $L = \mathcal{L}(A)$ .

$A$  heißt *Minimalautomat* für  $L$ , falls es keinen Automaten  $A'$  für  $L$  gibt, der weniger Zustände als  $A$  hat.

(Socher, S. 23)

### Anmerkungen & Fragen:

- Definieren sie ‚Minimalautomat‘ mithilfe der Äquivalenzrelation aus Definition 2.4

**Errata S. 23:** Wird im Startzustand des Automaten  $\dots$ , ist in diesem Fall  $\mathcal{L}(A_{LA}, z_3)$  nicht leer, sondern es gilt  $\mathcal{L}(A_{LA}, z_3) = \{\epsilon\}$ .

---

**Definition 2.8:** Isomorphie von Automaten

---

Die beiden Automaten  $A_1 = (Z_1, \Sigma, \delta_1, z_{01}, E_1)$  und  $A_2 = (Z_2, \Sigma, \delta_2, z_{02}, E_2)$  heißen *isomorph*, wenn es eine bijektive Abbildung  $f: Z_1 \rightarrow Z_2$  gibt mit:

$$f(z_{01}) = z_{02}$$

$$f(E_1) = E_2$$

$$z \rightarrow_a z' \text{ gdw. } f(z) \rightarrow_a f(z').$$

---

(Socher, S. 24)

**Errata S. 24:** Das Beispiel 2.3 bezieht sich auf die falsche Abbildung.

---

**Definition 2.9:** Äquivalenz von Zuständen

---

Zwei Zustände  $z$  und  $z'$  eines DEA  $A$  heißen *äquivalent*, in Zeichen  $z \equiv z'$ , falls  $\mathcal{L}(A, z) = \mathcal{L}(A, z')$  gilt.

---

(Socher, S. 25)

**Anmerkungen & Fragen:**

- Beweisen sie, dass die Relation  $\equiv$  eine Äquivalenzrelation auf der Zustandsmenge  $Z$  bildet.

---

**Definition 2.10:** Der Minimalautomat

---

Sei  $A = (Z, \Sigma, \delta, z_0, E)$  ein DEA. Sei  $[z]$  die Äquivalenzklasse von  $z$  bezüglich  $\equiv$ . Der Automat  $\bar{A} = (\bar{Z}, \Sigma, \bar{\delta}, \bar{z}_0, \bar{E})$  ist definiert durch:

$$\bar{Z} = \{[z] \mid z \in Z\},$$

$$\bar{\delta} = \{[z] \rightarrow_a [z'] \mid z \rightarrow_a z' \in \delta\},$$

$$\bar{z}_0 = [z_0],$$

$$\bar{E} = \{[z] \mid z \in E\}.$$

Dann ist  $\bar{A}$  der Minimalautomat für  $\mathcal{L}(A)$ .

---

(Socher, S. 26f)

**Anmerkungen & Fragen:**

- S.29 Minimierung eines DEA: Warum müssen zunächst die nichterreichbaren Zustände entfernt werden? Zeigen sie an einem Beispiel, was passiert, wenn dieser erste Schritt übersprungen wird.

---

**Definition 2.11:** Nichtdeterministischer endlicher Automat, NEA

---

Ein *nichtdeterministischer endlicher Automat* besteht aus den fünf Komponenten  $Z, \Sigma, \Delta, z_0, E$ . Dabei sind  $Z, \Sigma, z_0, E$  genauso definiert wie beim DEA (siehe Definition 2.1).

Für die Übergangsfunktion gilt  $\delta: Z \times \Sigma \rightarrow 2^Z$

---

(Socher, S. 30)

**Anmerkungen & Fragen:**

- Warum spricht man im Zusammenhang mit nichtdeterministischen endlichen Automaten auch von der Übergangsrelation?
-

---

**Definition 2.12:** Akzeptierung durch einen NEA

---

- a) Die Übergangsrelation  $\rightarrow$  auf der Menge der Konfigurationen des NEA  $A$  ist definiert durch

$$(z, aw) \rightarrow (z', w), \text{ falls } z \rightarrow_a z'$$

- b) Der NEA  $A$  akzeptiert das Eingabewort  $w$ , wenn es ein  $z_e \in E$  gibt mit

$$(z_0, w) \rightarrow^* (z_e, \varepsilon).$$

---

(Socher, S. 31)

---

**Definition 2.13:** NEA-DEA-Transformation

---

Sei  $A = (Z, \Sigma, \delta, z_0, E)$  ein NEA. Der DEA  $A' = (Z', \Sigma, \delta', z_0', E')$  ist folgendermaßen definiert:

■  $Z' = 2^Z$

■  $\delta' : Z' \times \Sigma \rightarrow Z'$  ist definiert durch:

$$\delta'(z', a) = \bigcup_{z \in z'} \delta(z, a)$$

■  $z_0' = \{z_0\}$

■  $E' = \{z \in Z' \mid z \cap E \neq \emptyset\}$

Dann gilt:  $A'$  ist äquivalent zu  $A$ .

---

(Socher, S. 35)

**Anmerkungen & Fragen:**

- Schreiben sie einen formalen Beweis, dass  $A$  und  $A'$  äquivalent sind. Orientieren sie sich an dem Beweis in Schöning (siehe Kurshomepage).
- Dieser Satz wird auch ‚Rabin-Scott-Theorem‘ genannt. Er wurde erstmals 1959 von Michael O. Rabin und Dana Scott in dem Aufsatz ‚Finite Automata and Their Decision Problems‘ veröffentlicht.

---

**Definition 2.14:** NEA mit  $\varepsilon$ -Übergängen (NEA/ $\varepsilon$ )

---

Ein *nichtdeterministischer endlicher Automat mit  $\varepsilon$ -Übergängen* (NEA/ $\varepsilon$ ) besteht aus den fünf Komponenten  $Z, \Sigma, \delta, z_0, E$ . Dabei sind  $Z, \Sigma, z_0, E$  genauso definiert wie beim DEA (siehe Definition 2.1).

Für die Übergangsfunktion gilt  $\delta : Z \times (\Sigma \cup \{\varepsilon\}) \rightarrow 2^Z$

---

(Socher, S. 36)

---

**Definition 2.15:** Akzeptierung durch einen NEA/ $\varepsilon$ 

---

Sei  $A$  ein NEA/ $\varepsilon$ . Die *Übergangsrelation*  $\rightarrow$  ist definiert durch

$$(z, aw) \rightarrow (z', w), \text{ falls } z \rightarrow_a z'$$

$$(z, w) \rightarrow (z', w), \text{ falls } z \rightarrow_\varepsilon z'$$

Der Automat  $A$  akzeptiert das Eingabewort  $w$  genau dann, wenn es ein  $z_e \in E$  gibt mit

$$(z_0, w) \rightarrow^* (z_e, \varepsilon)$$

---

(Socher, S. 37)

**Definition 2.16:** NEA/ $\epsilon$ -NEA-Transformation

Sei  $A = (Z, \Sigma, \delta, z_0, E)$  ein NEA/ $\epsilon$ . Der NEA  $A' = (Z, \Sigma, \delta', z_0, E')$  ist folgendermaßen definiert:

■  $\delta' : Z \times \Sigma \rightarrow Z$  ist definiert durch:

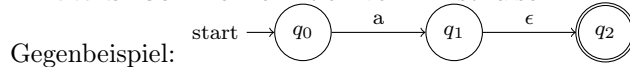
$$\delta'(z, a) = \bigcup_{z' \in [z]^*_\epsilon} \delta(z', a)$$

■  $E' = \bigcup_{z \in E} [z]^*_\epsilon$

Dann gilt:  $A'$  ist äquivalent zu  $A$ .

(Socher, S. 38)

**Errata S. 38:** Die Definition von  $E'$  ist falsch.

**Anmerkungen & Fragen:**

- Korrigieren sie die Definition von  $E'$ . Richtig wäre  $E' = \{z \in Z \mid \exists z_e \in E \text{ mit } z_e \in [z]^*\}$
- Beweisen sie, dass durch ihre Korrektur ein äquivalenter Automat entsteht.

**Definition 2.17:** Reguläre Sprache

Eine Sprache heißt *regulär*, wenn sie von einem endlichen Automaten (DEA, NEA oder NEA/ $\epsilon$ ) akzeptiert wird.

(Socher, S. 39)

**Hausaufgaben:**

Bearbeiten sie bitte die Aufgaben 2.1, 2.2, 2.3, 2.4, 2.6, 2.7c/e/g, 2.11, 2.13, 2.14, 2.17.

Sie können ihre Ergebnisse mit den Lösungsvorschlägen im Anhang abgleichen. Die Hausaufgaben werden nicht eingereicht. Zusätzlich sollten sie sich die Programme ‚machines‘ (<http://zeus.fh-brandenburg.de/~socher/tgi/>) und ‚exorciser‘ (<http://www.educ.ethz.ch/unt/um/inf/ti/exorciser/index>) installieren, die beide sehr gut für Übungszwecke geeignet sind.

**Das Ziege-Wolf-Kohlkopf-Problem:**

Ein Bauer möchte eine Ziege, einen Wolf und einen Kohlkopf zu einem Marktplatz bringen, der auf der anderen Seite eines Flusses liegt. Er hat ein kleines Boot, mit dem er jeweils nur entweder ein Tier oder den Kohlkopf mitnehmen kann. Der Wolf würde ohne Aufsicht sofort die Ziege fressen und die Ziege würde gerne den Kohlkopf fressen. Nur die Anwesenheit des Bauers verhindert, dass eines seiner ‚Produkte‘ gefressen wird. Wie muss der Bauer vorgehen, um alles sicher zum Markt zu bringen?

Konstruieren Sie einen endlichen Automaten, der das Problem modelliert und löst. Wenn Sie möchten, schreiben Sie zusätzlich ein Prolog-Programm für das Problem. Können Sie ein Prolog-Programm schreiben, das Ihnen alle minimalen Lösungen ausgibt?

Lösungen siehe Kurshomepage

**Mögliche Klausuraufgaben zu Kapitel 2:**

- Beweisen sie, dass die Relation  $\equiv$  in Definition 2.8 eine Äquivalenzrelation auf der Zustandsmenge  $Z$  bildet.
- Konstruieren Sie einen endlichen Automaten, der eine gegebene Sprache akzeptiert (vgl. Socher Aufgabe 2.7, kann sehr gut mit dem Programm ‚exorciser‘ geübt werden)

- Gegeben einen nichtdeterministischen endlichen Automaten, konstruieren sie einen äquivalenten deterministischen endlichen Automaten (vgl. Socher Aufgabe 2.17, kann sehr gut mit dem Programm ‚exorciser‘ geübt werden)
- Geben Sie eine rekursive Definition für  $|w|$  an (vgl. Socher Aufgabe 2.1 und 2.2)
- Beweisen Sie einen einfachen Zusammenhang (vgl. Socher Aufgabe 2.11 und 2.13)
- Gegeben ein  $NEA/\epsilon$ , konstruieren Sie einen äquivalenten NEA ohne epsilon-Übergänge (kann sehr gut mit dem Programm ‚exorciser‘ geübt werden)
- Konstruieren Sie zu einem gegebenen DEA einen äquivalenten DEA mit minimaler Zustandszahl (kann sehr gut mit dem Programm ‚exorciser‘ geübt werden)

### 3. Kapitel

**Definition 3.1:** Syntax und Semantik regulärer Ausdrücke

Die Menge der *regulären Ausdrücke* über einem Alphabet  $\Sigma$  ist folgendermaßen rekursiv definiert:

- $\emptyset$  und  $\epsilon$  sind reguläre Ausdrücke.
- Jedes Zeichen  $a \in \Sigma$  ist ein regulärer Ausdruck.
- Sind  $r$  und  $s$  reguläre Ausdrücke, so sind auch  $(rs)$ ,  $(r+s)$  und  $(r)^*$  reguläre Ausdrücke

Die von einem regulären Ausdruck  $r$  *dargestellte Sprache*  $\mathcal{L}(r)$  ist folgendermaßen definiert:

$$\begin{aligned} \mathcal{L}(\emptyset) &= \emptyset \\ \mathcal{L}(\epsilon) &= \{\epsilon\}. \\ \mathcal{L}(a) &= \{a\} \text{ für } a \in \Sigma \\ \mathcal{L}(rs) &= \mathcal{L}(r)\mathcal{L}(s) \\ \mathcal{L}(r+s) &= \mathcal{L}(r) \cup \mathcal{L}(s) \\ \mathcal{L}(r^*) &= \mathcal{L}(r)^* \end{aligned}$$

Zwei reguläre Ausdrücke  $r$  und  $s$  heißen *äquivalent*, in Zeichen  $r \equiv s$ , falls  $\mathcal{L}(r) = \mathcal{L}(s)$ .

(Socher, S. 46)

**Satz 3.1**

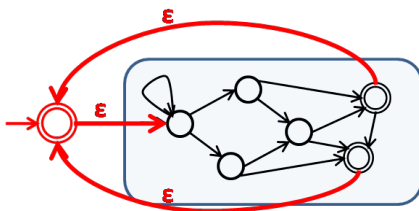
Sei  $\Sigma$  ein Alphabet. Ist  $r$  ein regulärer Ausdruck über  $\Sigma$ , so gibt es einen endlichen Automaten  $A$  über  $\Sigma$  mit  $\mathcal{L}(A) = \mathcal{L}(r)$ .

(Socher, S. 47)

**Errata S. 48:** Die Konstruktion in Abschnitt d in dem Beweis ist falsch.

**Anmerkungen & Fragen:**

- zeigen sie, dass die Konstruktion scheitert, wenn von  $z_0^1$  eine Schleife ausgeht (Beispiel:  $(1^*0)^*$ ).
- korrigieren sie die Konstruktion.  
Skizze zur korrigierten Konstruktion:



**Satz 3.2**

Sei  $\Sigma$  ein Alphabet. Ist  $A$  ein endlicher Automaten über  $\Sigma$ , so gibt es einen regulären Ausdruck  $r$  über  $\Sigma$  mit  $\mathcal{L}(A) = \mathcal{L}(r)$ .

(Socher, S. 49)

**Anmerkungen & Fragen:**

- Warum lässt sich jeder endliche Automaten in einen  $NEA/\epsilon$  in nur einem Endzustand transformieren?

**Errata S. 50:** Ein Wort ... ohne dass dazwischen ein Zustand mit einem Index größer  $-1$  durchlaufen wird.

---

**Satz 3.3**

---

Die Klasse der regulären Sprachen ist genau die Klasse der Sprachen, die von regulären Ausdrücken dargestellt werden können.

---

(Socher, S. 51)

**Errata S. 53:** Angenommen, es gäbe einen .... Wir wählen ein Wort ... dessen Länge mindestens  $N$  ist.

---

**Satz 3.4** Pumping-Lemma für reguläre Sprachen

---

Sei  $L$  eine reguläre Sprache. Dann existiert eine Zahl  $N \geq 0$ , sodass sich jedes Wort  $x \in L$  mit  $|x| \geq N$  in der folgenden Form schreiben lässt

$$x = uvw \text{ mit } |v| \geq 1 \text{ und } |uv| \leq N,$$

und für alle  $i \geq 0$  gilt, dass  $uv^i w \in L$  ist.

---

(Socher, S. 54)

**Anmerkungen & Fragen:**

- Für welche Art von Beweisen lässt sich das Pumpinglemma einsetzen?
- Warum wird im Pumpinglemma  $|v| \geq 1$  und  $|uv| < N$  gefordert?
- Hinweis zu S. 55 Mitte: Der Automat  $A_n$  hat tatsächlich  $2n + 2$  Zustände. Beachte, dass dem Automaten in Abbildung 3.5 der Fehlerzustand fehlt.

---

**Satz 3.5** (Satz von Myhill Nerode)

---

Sei  $L$  eine Sprache über  $\Sigma$ . Ist  $w \in \Sigma^*$ , so sei

$$\mathcal{L}(w) = \{v \in \Sigma^* \mid vw \in L\}.$$

Dann gilt:

- Ist  $L$  regulär, dann ist die Menge  $\{\mathcal{L}(w) \mid w \in \Sigma^*\}$  endlich.
  - Ist  $\{\mathcal{L}(w) \mid w \in \Sigma^*\}$  endlich, so ist  $L$  regulär.
  - In Fall b) gibt  $|\{\mathcal{L}(w) \mid w \in \Sigma^*\}|$  die Anzahl der Zustände eines minimalen DEA an, der  $L$  akzeptiert.
- 

(Socher, S. 57)

---

**Satz 3.6**

---

Seien  $L_1$  und  $L_2$  reguläre Sprachen über dem Alphabet  $\Sigma$ . Dann sind auch die Sprachen  $L_1 \cup L_2$ ,  $L_1 L_2$ , sowie  $L_1^*$  regulär.

---

(Socher, S. 60)

---

**Satz 3.7**

---

Sind  $L_1$  und  $L_2$  reguläre Sprachen über dem Alphabet  $\Sigma$ , so sind auch die Sprachen  $\overline{L_1}$  und  $L_1 \cap L_2$  regulär

---

(Socher, S. 61)

---



**Satz 3.8**

---

Die Klasse der regulären Sprachen ist abgeschlossen unter den Mengenoperationen Vereinigung, Durchschnitt, Komplement, Konkatenation und Kleene-Stern.

---

(Socher, S. 61)

**Hausaufgaben:**

Bearbeiten sie bitte die Aufgaben 3.2,3.3,3.4,3.5,3.6,3.8,3.10,3.11,3.12,3.13,3.14,3.15

Sie können ihre Ergebnisse mit den Lösungsvorschlägen im Anhang abgleichen. Die Hausaufgaben werden nicht eingereicht.

**Mögliche Klausuraufgaben zu Kapitel 3:**

- Konstruktion eines regulären Ausdrucks zu einer Sprache (Aufgabe 3.2)
- Bestimmung von Wörtern einer durch einen regulären Ausdrücke beschriebenen Sprache (Aufgabe 3.4)
- Prüfen, ob eine Sprache regulär ist (Aufgabe 3.8, 3.10)
- Aussagen über reguläre Ausdrücke (Aufgabe 3.11, 3.12)
- Einfache Beweise wie Aufgabe 3.14 und 3.16

## 4. Kapitel

### Anmerkungen & Fragen:

- Von welchem Typ ist die Grammatik auf Seite 66?
- Zu welcher Sprachklasse gehört die von der Grammatik generierte Sprache?
- Muss Grammatiktyp und Sprachklasse immer übereinstimmen?
- Was ist  $V$ ,  $\Sigma$ ,  $S$  dieser Grammatik (siehe Def. 4.1)

---

#### Definition 4.1: Grammatik

---

Eine *Grammatik* besteht aus den vier Komponenten  $V, \Sigma, P, S$ .

- $V$  ist eine endliche Menge von *Variablen*.
  - $\Sigma$  ist ein Alphabet, das *Terminalalphabet*.
  - $P$  ist eine Menge von *Produktionen* (auch *Regeln* genannt) der Form  $l \rightarrow r$ , wobei  $l \in (V \cup \Sigma)^+$  und  $r \in (V \cup \Sigma)^*$ .
  - $S \in V$  ist die *Startvariable*.
- 

(Socher, S. 66)

### Anmerkungen & Fragen:

- Welche Einschränkungen bestehen für  $l$  und  $r$

---

#### Definition 4.2: Ableitungsrelation, erzeugte Sprache, Äquivalenz von Grammatiken

---

Sei  $G = (V, \Sigma, P, S)$  eine Grammatik.

Die *Ableitungsrelation*  $\Rightarrow_G$  auf der Menge  $(V \cup \Sigma)^*$  ist definiert durch

$$vlu \Rightarrow_G vru$$

wobei  $v, u \in (V \cup \Sigma)^*$  und  $l \rightarrow r \in P$ .

Die von  $G$  *erzeugte Sprache*  $\mathcal{L}(G)$  ist definiert durch

$$\mathcal{L}(G) = [S]^* \cap \Sigma^* = \{w \in \Sigma^* \mid S \Rightarrow_G^* w\}.$$

Zwei Grammatiken  $G_1$  und  $G_2$  heißen *äquivalent*, wenn sie dieselbe Sprache erzeugen, das heißt, wenn  $\mathcal{L}(G_1) = \mathcal{L}(G_2)$  gilt.

---

(Socher, S. 67)

### Anmerkungen & Fragen:

- $[S]^* \subseteq (V \cup \Sigma)^*$  steht für die Menge aller aus  $S$  ableitbaren Wörter.
- $[S]^* \cap \Sigma^*$  steht also für die Menge aller aus  $S$  ableitbaren Wörter, die nur aus Terminalen bestehen.

### Anmerkungen & Fragen:

- zu S. 69: Wieso werden rechtslineare Grammatiken rechtslinear genannt?

---

#### Satz 4.1 Transformation einer rechtslinearen Grammatik in einen NEA

---

Sei  $G = (V, \Sigma, P, S)$  eine rechtslineare Grammatik. Der NEA  $A = (Z, \Sigma, \delta, z_0, E)$  sei definiert durch:

$$Z = V, z_0 = S, E = \{A \mid A \rightarrow \varepsilon \in P\}, \delta = \{A \rightarrow_a B \mid A \rightarrow aB \in P\}.$$

Dann gilt:  $\mathcal{L}(A) = \mathcal{L}(G)$ .

---

**Satz 4.2** Transformation eines NEA in eine rechtslineare Grammatik

Sei  $A = (Z, \Sigma, \delta, z_0, E)$  ein NEA. Die rechtslineare Grammatik  $(V, \Sigma, P, S)$  sei definiert durch:

$$V = Z, S = z_0, P = \{z \rightarrow az' \mid z \rightarrow_a z' \in \delta\} \cup \{z \rightarrow \varepsilon \mid z \in E\}.$$

Dann gilt:  $\mathcal{L}(G) = \mathcal{L}(A)$ .

(Socher, S. 71)

**Anmerkungen & Fragen:**

- Warum stören eigentlich Fehlerzustände des Automaten bei der Transformation nicht? Zu was werden sie?

**Satz 4.3** Äquivalenz von regulären Sprachen und Typ-3-Sprachen

Die Klasse der regulären Sprachen ist die Klasse der Sprachen, die von regulären Grammatiken erzeugt werden können.

(Socher, S. 71)

**Tabelle 4.1:** Entscheidungsprobleme für reguläre Sprachen

Problem	Gegeben	Gefragt	Entscheidbar?
Wortproblem	$L$ und $w \in \Sigma^*$	Gilt $w \in L$ ?	Ja
Leerheitsproblem	$L$	Gilt $L = \emptyset$ ?	Ja
Endlichkeitsproblem	$L$	Gilt $ L  < \infty$ ?	Ja
Äquivalenzproblem	$L_1$ und $L_2$	Gilt $L_1 = L_2$ ?	Ja

(Socher, S. 73)

**Anmerkungen & Fragen:**

- Wie würden Sie die Entscheidungsprobleme entscheiden? Wie würden Sie die Entscheidungsprobleme implementieren?

**Definition 4.3:** Kontextfreie Grammatik

Eine Grammatik heißt *kontextfrei*, wenn ihre Regeln von der Form  $l \rightarrow r$ , mit  $l \in V$  sind.

(Socher, S. 74)

**Definition 4.4:** Chomsky-Normalform (CNF)

Eine kontextfreie Grammatik  $G$  mit  $\varepsilon \notin \mathcal{L}(G)$  heißt in *Chomsky-Normalform*, wenn alle ihre Regeln von einer der beiden folgenden Formen sind:

$$A \rightarrow BC \text{ oder}$$

$$A \rightarrow a$$

mit  $A, B, C \in V, a \in \Sigma$ .

(Socher, S. 78)

**Errata S. 80:** unten: Als Eingabewort ... aus welchen Variablen sich das Teilwort  $a_j$  der Länge ...

**Errata S. 81:** Vorsicht, in dem Algorithmus sind die Rollen vertauscht. Im Fließtext auf Seite 80 und 81 steht  $T[i, j]$  für die Menge der Variablen, aus denen sich das Wort der Länge  $i$ , das an der Position  $j$  beginnt, ableiten lässt. Im Algorithmus auf Seite 81 steht  $T[i, j]$  für die Menge der Variablen, aus denen sich das Wort der Länge  $j$ , das an der Position  $i$  beginnt, ableiten lässt.

---

**Satz 4.4** Pumping-Lemma für kontextfreie Sprachen

---

Sei  $L$  eine kontextfreie Sprache. Dann existiert eine Zahl  $N \geq 0$ , sodass sich jedes Wort  $z \in L$  mit  $|z| \geq N$  in der folgenden Form schreiben lässt

$$z = uvwxy \text{ mit } |vx| \geq 1 \text{ und } |vwx| \leq N,$$

und für alle  $i \geq 0$  gilt, dass  $uv^iwx^iy \in L$  ist.

---

(Socher, S. 82)

**Anmerkungen & Fragen:**

- Lesen Sie unbedingt das Beispiel 4.6, das zeigt, wie das Pumping-Lemma angewandt wird.

---

**Satz 4.5** Abgeschlossenheitseigenschaften kontextfreier Sprachen

---

Die Klasse der kontextfreien Sprachen ist abgeschlossen unter Vereinigung, Konkatination und Kleene-Stern. Sie ist nicht abgeschlossen unter Durchschnitt und Komplement.

---

(Socher, S. 83)

**Errata S. 83:** Dies ist zwar kein echter Fehler, aber es fehlt der wichtige Satz, dass kontextfreie Sprachen abgeschlossen sind unter dem Schnitt mit regulären. Also, wenn  $L_k$  eine beliebige kontextfreie Sprache ist und  $L_r$  eine beliebige reguläre, dann ist  $L_k \cap L_r$  eine kontextfreie Sprache.

**Errata S. 85:** Zur Lösung ... Eine Kante verläuft von ...  $V_2$  auf der rechten Seite derselben Regel  $R$ .

**Errata S. 85:** Beispiel 4.7: Der Pfeil von A nach S muss in beiden Abbildungen umgedreht werden

---

**Definition 4.5:** Nichtdeterministischer Kellerautomat (NKA)

---

Ein *nichtdeterministischer Kellerautomat* (NKA) besteht aus den Komponenten  $Z, \Sigma, \Gamma, \perp, \Delta, z_0, E$ :

- $Z$  ist eine endliche Menge von Zuständen
  - $\Sigma$  ist ein Alphabet, das sog. Eingabealphabet
  - $\Gamma$  ist ein Alphabet, das sog. Kelleralphabet
  - $\perp \in \Gamma$  ist das Keller-Startsymbol
  
  - $\Delta \subseteq Z \times \Gamma \times (\Sigma \cup \{\epsilon\}) \times Z \times \Gamma^*$  ist die Übergangsrelation
  - $z_0 \in Z$  ist der Startzustand
  - $E \subseteq Z$  ist die Menge der Endzustände
- 

(Socher, S. 87f)

**Errata S. 87:** b) Tabelle 4.4 ... bei der Verarbeitung des Wortes  $w = 00101$

---

**Definition 4.6:** Akzeptierung mit Endezustand

---

Sei  $K$  ein NKA.

Die Relation  $\rightarrow$  auf der Menge der Konfigurationen von  $K$  ist definiert durch

$$(z, AW, aw) \rightarrow (z', VW, w), \text{ falls } (z, A) \rightarrow_a (z', V)$$

$$(z, AW, w) \rightarrow (z', VW, w), \text{ falls } (z, A) \rightarrow_\varepsilon (z', V).$$

Der Automat  $K$  akzeptiert das Eingabewort  $w$  mit Endezustand genau dann, wenn es ein  $z_e \in E$  und  $W \in \Gamma^*$  gibt mit

$$(z_0, \perp, w) \rightarrow^* (z_e, W, \varepsilon).$$

Die vom Automaten  $K$  mit Endezustand akzeptierte Sprache  $\mathcal{L}_E(K)$  ist definiert durch:

$$\mathcal{L}_E(K) = \{w \in \Sigma^* \mid K \text{ akzeptiert } w \text{ mit Endezustand}\}.$$

---

(Socher, S. 89)

**Definition 4.7:** Akzeptierung mit leerem Keller

---

Der Kellerautomat  $K$  akzeptiert das Eingabewort  $w$  mit leerem Keller, wenn

$$(z_0, \perp, w) \rightarrow^* (z, \varepsilon, \varepsilon)$$

gilt, das heißt, wenn zum Ende der Berechnung der Keller leer ist.

Die von  $K$  mit leerem Keller akzeptierte Sprache  $\mathcal{L}_{LK}(K)$  ist definiert durch:

$$\mathcal{L}_{LK}(K) = \{w \in \Sigma^* \mid K \text{ akzeptiert } w \text{ mit leerem Keller}\}$$

---

(Socher, S. 89)

**Errata S. 91:** Der Automat rät, ... erreicht ist. Das oberste Kellerzeichen ist in diesem Fall das Zeichen, das genau in der Wortmitte gestanden hat.

**Satz 4.6**

---

Zu jeder kontextfreien Grammatik  $G$  gibt es einen äquivalenten NKA  $K_G$ .

---

(Socher, S. 94)

**Satz 4.7** Äquivalenz von Kellerautomaten und kontextfreien Grammatiken

---

Die Klasse der kontextfreien Sprachen ist genau die Klasse der Sprachen, die von einem Kellerautomaten akzeptiert werden.

---

(Socher, S. 95)

**Definition 4.8:** Deterministischer Kellerautomat, deterministisch kontextfreie Sprachen

---

Ein Kellerautomat heißt *deterministisch* (DKA), falls für alle  $z \in Z$ ,  $A \in \Gamma$ ,  $a \in \Sigma$  gilt:

$$|\delta(z, A, a)| + |\delta(z, A, \varepsilon)| \leq 1$$

Eine Sprache  $L$  heißt *deterministisch kontextfrei*, falls es einen deterministischen Kellerautomaten gibt, der  $L$  mit Endezustand akzeptiert.

---

(Socher, S. 95)

**Anmerkungen & Fragen:**

- Beachte, dass die Menge der deterministisch kontextfreien Sprachen eine echte Teilmenge der kon-

textfreien Sprachen ist.

**Definition 4.9:** Kontextsensitive (Typ-1-)Grammatik

Eine Grammatik heißt *kontextsensitiv (oder vom Typ 1)*, wenn ihre Regeln von der folgenden Form sind:

$$l \rightarrow r, \text{ mit } |l| \leq |r|.$$

Als einzige Ausnahme ist die Regel  $S \rightarrow \epsilon$  zulässig. Enthält die Grammatik diese Regel, so darf jedoch die Startvariable  $S$  nicht auf der rechten Seite einer Regel vorkommen.

Eine Sprache  $L$  heißt *kontextsensitiv (oder vom Typ 1)*, wenn sie sich von einer *kontextsensitiven* Grammatik erzeugen lässt.

(Socher, S. 96)

**Definition 4.10:** Grammatik vom Typ 0

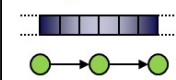
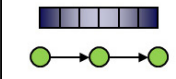


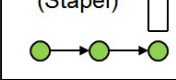





Eine allgemeine Grammatik (ohne Einschränkungen an die Syntax der Regeln) heißt auch *Grammatik vom Typ 0 oder Semi-Thue-System*.

Eine Sprache  $L$  heißt *vom Typ 0*, wenn sie sich von einer *Typ-0-Grammatik* erzeugen lässt.

(Socher, S. 97)

$$\text{TYP-3}_\Sigma \subset \text{TYP-2}_\Sigma \subset \text{TYP-1}_\Sigma \subset \text{TYP-0}_\Sigma \subset 2^{\Sigma^*}$$

Dabei ist  $2^{\Sigma^*}$  die Klasse aller Sprachen über  $\Sigma$ .

Sprache	Automat	Grammatik	Erkennung	Abhängigkeit
rekursiv aufzählbar	Turing Maschine 	unbeschränkt $Baa \rightarrow \epsilon$	unentscheidbar	beliebig
kontext- sensitiv	linear gebunden 	kontext- sensitiv $\gamma A \delta \rightarrow \gamma \beta \delta$	NP-vollständig 	überkreuzt 
kontext- frei	Kellerautomat (Stapel) 	kontextfrei $C \rightarrow bABa$	polynomiell 	eingebettet 
regulär	endlicher Automat 	regulär $A \rightarrow bA$	linear 	strikt lokal 

	Typ3	Typ2	Typ1	Typ0
Wortproblem	E	E	E	U
Leerheitsproblem	E	E	U	U
Äquivalenzproblem	E	U	U	U

E steht für entscheidbar  
U steht für unentscheidbar

## Hausaufgaben:

Bearbeiten sie bitte die Aufgaben 4.1, 4.6,4.10,4.11, 4.12, 4.13, 4.15

Sie können ihre Ergebnisse mit den Lösungsvorschlägen im Anhang abgleichen. Die Hausaufgaben werden nicht eingereicht.

## Mögliche Klausuraufgaben zu Kapitel 4:

- Geben Sie zu einer Sprache eine rechtslineare Grammatik an (Aufgabe 4.1)
- Gegeben ein endlicher Automat, geben sie eine rechtslineare Grammatik an, die die Sprache generiert, die der Automat akzeptiert.
- Gegeben eine rechtslineare Grammatik, geben sie einen endlichen Automaten an, der die Sprache akzeptiert, die die Grammatik generiert.
- Beweisen Sie, dass die Klasse der kontextfreien Sprachen abgeschlossen ist unter Konkatenation (S. 83).
- Wandeln Sie eine Grammatik in CNF um (Aufgabe 4.10).
- Zeigen Sie, dass eine Sprache nicht kontextfrei ist(Aufgabe 4.11).
- Konstruieren Sie einen Kellerautomaten zu einer gegebenen Sprache und stellen Sie die Verarbeitung eines gegebenen Wortes dar (Aufgabe 4.15). Gibt es einen deterministischen Kellerautomaten zu der Sprache?
- Geben Sie zu einer gegebenen kontextfreien Grammatik einen korrespondierenden Kellerautomaten an.
- Geben Sie zu einem gegebenen Kellerautomaten eine korrespondierende kontextfreie Grammatik an.

## 5. Kapitel

---

### Definition 5.1: Turing-Maschine

---

Eine (*deterministische*) *Turing-Maschine* (TM) besteht aus den Komponenten  $Z, \Sigma, \Gamma, \delta, z_0, \#, E$ .

- $Z$  ist eine endliche Menge von *Zuständen*.
  - $\Sigma \subset \Gamma$  ist das *Eingabealphabet*.
  - $\Gamma$  ist das *Bandalphabet*.
  - $\delta : Z \times \Gamma \rightarrow Z \times \Gamma \times \{L, R\}$  ist die *Übergangsfunktion*.  
 $\delta$  ist eine partielle Funktion. Wir schreiben  $\delta(z, a) = \perp$ , falls  $\delta(z, a)$  undefiniert ist.
  - $z_0 \in Z$  ist der *Startzustand*.
  - $\# \in \Gamma - \Sigma$  ist das *Leerzeichen* (auch *Blanksymbol* genannt).
  - $E \subseteq Z$  ist die Menge der *Endezustände*.
- 

(Socher, S. 104)

### Definition 5.2: Konfiguration einer TM

---

Eine *Konfiguration einer Turing-Maschine* ist ein Tripel  $(v, z, w)$  mit  $v, w \in \Gamma^*$  und  $z \in Z$ . Dabei ist  $z$  der aktuelle Zustand,  $v$  der links vom Zeiger stehende Bandinhalt und  $w$  der rechts vom Zeiger stehende Bandinhalt. Der Zeiger steht dabei auf dem ersten Zeichen von  $w$ .

Die *Übergangsrelation* auf der Menge der Konfigurationen ist folgendermaßen definiert:

$$(v, z, aw) \rightarrow (vb, z', \overline{w}), \text{ falls } \delta(z, a) = (z', b, R) \text{ und}$$

$$(vc, z, aw) \rightarrow (\overline{v}, z', cbw), \text{ falls } \delta(z, a) = (z', b, L).$$

Dabei ist  $\overline{w}$  für  $w \in \Gamma^*$  definiert durch:

$$\overline{w} = \begin{cases} w & \text{falls } w \neq \varepsilon \\ \# & \text{falls } w = \varepsilon \end{cases}$$

---

(Socher, S. 106)

**Errata S. 106:** Definition 5.2: Dabei ist  $z$  der ... und  $w$  der am Zeiger beginnende Bandinhalt.

### Anmerkungen & Fragen:

- Auf <http://www.matheprisma.uni-wuppertal.de/Module/Turing/index.htm> finden Sie ein sehr gutes Selbstlernmodul für Turingmaschinen. Wenn Sie dieses Durcharbeiten sind Sie bestens vorbereitet auf die kommende Sitzung.
-



---

**Definition 5.3:** Akzeptierung durch eine TM, rekursiv aufzählbare und rekursive Sprachen

---

Die Turing-Maschine  $M$  akzeptiert das Eingabewort  $w$  genau dann, wenn es ein  $z_e \in E$  sowie Wörter  $v$  und  $u \in \Gamma^*$  gibt mit

$$(\#, z_0, w) \rightarrow^* (v, z_e, u).$$

Die von  $M$  akzeptierte Sprache  $\mathcal{L}(M)$  ist definiert durch:

$$\mathcal{L}(M) = \{w \in \Sigma^* \mid M \text{ akzeptiert } w\}.$$

Eine Sprache  $L$  heißt *rekursiv aufzählbar*, wenn es eine Turing-Maschine  $M$  gibt, die  $L$  akzeptiert.

Eine Sprache  $L$  heißt *rekursiv*, wenn es eine Turing-Maschine  $M$  gibt, die  $L$  akzeptiert und die für jede Eingabe terminiert.

---

(Socher, S. 106)

**Errata S. 107:**  $\text{delta}(Z, A, Z_1, B, X) \dots$  beschreibt die Funktion bzw. Relation  $\delta$ .

**Errata S. 107:** Die Akzeptanz eines Wortes durch die Turingmaschine ist definiert durch die folgenden Prolog-Prädikate:

---

---

**Satz 5.1** Äquivalenz von deterministischen und nichtdeterministischen Turing-Maschinen

---

Die Klasse der von deterministischen Turing-Maschinen akzeptierten Sprachen ist gleich der Klasse der von nichtdeterministischen Turing-Maschinen akzeptierten Sprachen.

---

(Socher, S. 110)

---

**Satz 5.2**

---

Zu jeder Sprache  $L$  vom Typ 0 über einem Alphabet  $\Sigma$  gibt es eine Turing-Maschine, die  $L$  akzeptiert.

---

(Socher, S. 111)

**Errata S. 111:** Beachte, dass  $A$  die Produktionen  $u \rightarrow v$  zwar nichtdeterministisch wählt, dabei aber Backtracking einsetzt, um alle Möglichkeiten auszuschöpfen, um zur Akzeptanz eines Wortes zu gelangen (siehe Prolog-Code).

---

---

**Satz 5.3**

---

Ist  $A$  eine Turing-Maschine über  $\Sigma$ , dann ist  $\mathcal{L}(A)$  vom Typ 0.

---

(Socher, S. 112)

---

**Satz 5.4** Äquivalenz von Turing-Maschinen und Typ-0-Grammatiken

---

Die Klasse der rekursiv aufzählbaren Sprachen ist gleich der Klasse der Sprachen vom Typ 0.

---

(Socher, S. 113)

---

**Satz 5.5**

Zu jeder Sprache  $L$  vom Typ 1 gibt es einen LBA, der  $L$  akzeptiert. Ferner gibt es eine TM, die das Wortproblem für  $L$  entscheidet. Umgekehrt gilt auch:  
Ist  $A$  ein linear beschränkter Automat mit  $\varepsilon \notin \mathcal{L}(A)$ , so ist  $\mathcal{L}(A)$  vom Typ 1.

(Socher, S. 114)

**Satz 5.6** Äquivalenz von LBA und kontextsensitiven Grammatiken

Die Klasse der von linear beschränkten Automaten akzeptierten Sprachen, die das leere Wort nicht enthalten, ist gleich der Klasse der Typ-1-Sprachen.  
Das Wortproblem für Typ-1-Sprachen ist entscheidbar.

(Socher, S. 114)

**Definition 5.4:** Turing-Berechenbarkeit für Funktionen auf  $\Sigma^*$ 

Die Turing-Maschine  $M$  berechnet die Funktion  $f: \Sigma^* \rightarrow \Sigma^*$ , falls

$$f(w) = v \text{ genau dann wenn } (\#, z_0, w) \rightarrow^* (\#, z_e, v)$$

mit  $z_e \in E$ . Die Funktion  $f$  heißt in diesem Fall Turing-berechenbar. Wir schreiben  $f_M$  für die von der TM  $M$  berechnete Funktion.

(Socher, S. 115)

**Definition 5.5:** Turing-Berechenbarkeit für Funktionen auf  $\mathbb{N}^k$ 

Die Turing-Maschine  $M$  berechnet die Funktion  $f: \mathbb{N}^k \rightarrow \mathbb{N}$ , falls

$$(\#, z_0, u(n_1, n_2, \dots, n_k)) \rightarrow^* (\#, z_e, u(f(n_1, n_2, \dots, n_k)))$$

mit  $z_e \in E$ , falls  $f(n_1, n_2, \dots, n_k)$  definiert ist. Die Funktion  $f$  heißt in diesem Fall Turing-berechenbar oder auch partiell rekursiv. Ist  $f$  eine totale Funktion, so heißt  $f$  auch rekursiv.

(Socher, S. 115f)

**Tabelle 5.2:** Übersicht über die Sprachklassen

Sprach- klasse	Name	akzeptierender Automat	erzeugende Grammatik
Typ 0	rekursiv aufzählbar	Turing-Maschinen	beliebige Grammatik
Typ 1	kontextsensitiv	linear beschränkte Turing-Maschinen	kontextsensitive Gram- matik
Typ 2	kontextfrei	nichtdeterministische Kellerautomaten	kontextfreie Grammatik
DKF	deterministisch kontextfrei	deterministische Kellerautomaten	LR(k)-Grammatiken
Typ 3	regulär	endliche Automaten	reguläre Grammatik

(Socher, S. 115)

**Tabelle 5.3:** Abgeschlossenheitseigenschaften der Sprachklassen

Sprach- klasse	Durchschnitt	Vereinigung	Komplement	Konkatenation	Kleene-Stern
Typ 0	ja	ja	nein	ja	ja
Typ 1	ja	ja	ja	ja	ja
Typ 2	nein	ja	nein	ja	ja
DKF	nein	nein	ja	nein	nein
Typ 3	ja	ja	ja	ja	ja

(Socher, S. 115)

**Tabelle 5.4:** Entscheidbarkeitsresultate für die Sprachklassen

Sprach- klasse	Wortproblem	Leerheitsproblem	Endlichkeits- problem	Äquivalenzproblem
Typ 0	nein	nein	nein	nein
Typ 1	ja	nein	nein	nein
Typ 2	ja	ja	ja	nein
DKF	ja	ja	ja	?
Typ 3	ja	ja	ja	ja

(Socher, S. 116)

**Definition 5.6:** Loop-Berechenbarkeit

Das LOOP-Programm  $P$  mit  $|V(P)| = r$  berechnet die Funktion  $f: \mathbb{N}^k \rightarrow \mathbb{N}$ , mit  $k < r$  falls für alle  $\mathbf{n} \in \mathbb{N}^k$

$$\delta((0, \mathbf{n}, \mathbf{0}), P) = \mathbf{v} \text{ mit } \mathbf{v}_1 = f(\mathbf{n}).$$

Die Funktion  $f$  heißt in diesem Fall LOOP-berechenbar.

(Socher, S. 121)

**Satz 5.7**

Die Funktion  $a_k$  lässt sich durch ein LOOP-Programm mit genau  $k$  LOOP-Anweisungen berechnen, jedoch nicht durch ein LOOP-Programm mit weniger LOOP-Anweisungen.

(Socher, S. 122)

**Definition 5.7:** While-Berechenbarkeit

Das While-Programm  $P$  mit  $|V(P)| = r$  berechnet die (partielle) Funktion  $f: \mathbb{N}^k \rightarrow \mathbb{N}$ , mit  $k < r$  falls

$$\delta((0, \mathbf{n}, \mathbf{0}), P) = \mathbf{v} \text{ mit } \mathbf{v}_1 = f(\mathbf{n})$$

für alle  $\mathbf{n} = (n_1, \dots, n_k) \in \mathbb{N}^k$ , für die  $f(\mathbf{n})$  definiert ist. Die Funktion  $f$  heißt in diesem Fall *While-berechenbar*.

(Socher, S. 124)

---

**Satz 5.8** Äquivalenz von Turing-Berechenbarkeit und While-Berechenbarkeit

---

Eine (partielle) Funktion ist Turing-berechenbar genau dann, wenn sie While-berechenbar ist.

---

(Socher, S. 125)

---

**Definition 5.8:** Goto-Berechenbarkeit

---

Das GOTO-Programm  $P$  mit  $|V(P)| = r$  berechnet die (partielle) Funktion  $f: \mathbb{N}^k \rightarrow \mathbb{N}$ , mit  $k < r$ , falls

$$((0, \mathbf{n}, \mathbf{0}), 1) \rightarrow^* (\mathbf{v}, 0) \text{ mit } \mathbf{v}_1 = f(\mathbf{n})$$

für alle  $\mathbf{n}$ , für die  $f(\mathbf{n})$  definiert ist. Die Funktion  $f$  heißt in diesem Fall GOTO-berechenbar.

---

(Socher, S. 126)

---

## Codierung von Turing-Maschinen

Damit die universelle Turing-Maschine  $U$  die Maschinentabelle  $M$  einer konkreten Turing-Maschine simulieren kann, muss  $M$  zunächst mittels eines geeigneten Codierungsverfahrens in Binärcode codiert werden, sodass dieser als Eingabe von  $U$  gelesen werden kann. Zur Vereinfachung treffen wir einige einschränkende Annahmen über die Parameter der Maschine  $M$ . Die Zeichen des Eingabe- und des Bandalphabets der Maschine  $M$  lassen sich unter ausschließlicher Verwendung der drei Zeichen 0, 1 und # codieren. Weiterhin nehmen wir an, dass  $Z = \{z_1, z_2, \dots, z_n\}$  die Zustandsmenge von  $M$  ist mit  $z_1$  als Startzustand und  $z_2$  als einzigem Endezustand, denn jede Turing-Maschine mit mehreren Endezuständen lässt sich in eine äquivalente Maschine mit nur einem Endezustand transformieren.  $M$  ist also eine normierte Maschine mit

- Zustandsmenge  $Z = \{z_1, z_2, \dots, z_n\}$ ,
- Eingabealphabet  $\Sigma = \{0, 1\}$ ,
- Bandalphabet  $\Gamma = \{0, 1, \#\}$ ,
- Startzustand  $z_1$  und
- einzigem Endezustand  $z_2$ .

Wir definieren

$$X_1 = 0, X_2 = 1, X_3 = \#$$

für die Zeichen des Bandalphabets sowie

$$R_1 = L, R_2 = R$$

für die Richtungen links bzw. rechts. Die Turing-Anweisung

$$\delta(z_i, X_j) = (z_k, X_b, R_m)$$

wird dann durch folgendes Binärwort codiert:

$$0^i 10^j 10^k 10^l 10^m$$

Beispielsweise wird die Anweisung  $\delta(z_3, 0) = (z_4, 1, L)$  einer normierten Maschine zunächst in der Form  $\delta(z_3, X_1) = (z_4, X_2, R_1)$  notiert und anschließend als Binärwort 000101000010010 codiert.

Sind nun  $c_1, c_2, \dots, c_r$  die Codierungen der Anweisungen der Maschine  $M$ , so wird die komplette Maschinentabelle von  $M$  dargestellt durch das Binärwort

$$111c_111c_211 \dots 11c_r111.$$

(Socher, S. 128f)

**Errata S. 130:** Gibt es dagegen einen Block der Form  $110^i 10^j 1 \dots$

**Errata S. 131:**  $0^k 10^l 10^m 11$  ist und ersetzt das ...

## Hausaufgaben:

Bearbeiten sie bitte die Aufgaben 5.1, 5.2, 5.3, 5.5. Entwickeln Sie eine Turingmaschine, die den Bandinhalt kopiert.

Sie können ihre Ergebnisse mit den Lösungsvorschlägen im Anhang abgleichen. Die Hausaufgaben werden nicht eingereicht.

## Mögliche Klausuraufgaben zu Kapitel 5:

- Geben Sie zu einer Sprache eine Turingmaschine an, die die Sprache akzeptiert (Aufgabe 5.2,5.3,5.4)
- Führen Sie die Berechnung für ein Beispielwort durch.
- Multiple-Choice Questions zu Aussagen über Berechenbarkeit (welche der folgenden Aussagen ist wahr? Bsp. Alle berechenbaren Funktionen sind rekursiv. Sind alle rekursiv aufzählbaren Sprachen auch regulär?)
- Geben Sie zu einer gegebenen Turingmaschine die Gödelnummer an.
- Wandeln sie eine Gödelnummer in die entsprechende Turingmaschine um.

## 6. Kapitel

---

### Definition 6.1: Entscheidbarkeit

---

Sei  $\Sigma$  ein Alphabet. Die Sprache  $L \subseteq \Sigma^*$  heißt *entscheidbar*, falls die charakteristische Funktion  $\chi : \Sigma^* \rightarrow \{0, 1\}$ ,  $\chi(x) = [x \in L]$  Turing-berechenbar ist.

---

(Socher, S. 136)

---

### Satz 6.1

---

Die Sprache  $L$  ist entscheidbar genau dann, wenn sie rekursiv ist, das heißt, wenn es eine Turing-Maschine gibt, die stets hält und  $L$  akzeptiert.

---

(Socher, S. 136)

---

### Definition 6.2: Semi-Entscheidbarkeit

---

Die Sprache  $L \subseteq \Sigma^*$  heißt *semi-entscheidbar*, falls es eine Turing-Maschine  $M$  gibt mit:

$$f_M(w) = \begin{cases} 1 & \text{falls } w \in L \\ \perp & \text{falls } w \notin L \end{cases}$$

Dies ist genau dann der Fall, wenn  $L$  rekursiv aufzählbar ist.

---

(Socher, S. 137f)

---

### Anmerkungen & Fragen:

- Zur Erinnerung  $\perp$  steht für *nicht definiert*.
- 

### Satz 6.2

---

Die Sprache  $L \neq \emptyset$  ist semi-entscheidbar genau dann, wenn es eine berechenbare totale Funktion  $f : \mathbb{N} \rightarrow L$  gibt mit  $L = f(\mathbb{N})$ .

---

(Socher, S. 138)

---

### Satz 6.3

---

Die Sprache  $L \subseteq \Sigma^*$  ist entscheidbar genau dann, wenn sowohl  $L$  als auch  $\bar{L} = \Sigma^* - L$  semi-entscheidbar sind.

---

(Socher, S. 139)

---

**Errata S. 141:** Warum ist es so klar, dass der Code  $\beta$  aus Beispiel 6.1 nicht geeignet ist?

---

### Definition 6.3: Postsches Korrespondenzproblem

---

Das *Postsche Korrespondenzproblem* (PKP) lautet:

Gegeben: Alphabet  $\Sigma$  und  $\Gamma$  und zwei Funktionen  $\alpha, \beta : \Sigma \rightarrow \Gamma^*$

Gefragt: Gibt es ein Wort  $w \in \Sigma^* - \{\varepsilon\}$  mit  $\alpha(w) = \beta(w)$ ?

---

(Socher, S. 143)

---

---

**Definition 6.4:** Allgemeines Halteproblem

---

Das allgemeine Halteproblem lautet:

Gegeben: eine Turing-Maschine  $M$  und ein Eingabewort  $w$ .

Gefragt: Terminiert  $M$  bei Eingabe von  $w$ ?

---

(Socher, S. 145)

**Errata S. 143:** Es ist jedoch leicht zu sehen, dass ... (siehe Aufgabe 6.4).

---

**Definition 6.5:** Spezielles Halteproblem

---

Das spezielle Halteproblem lautet:

Gegeben: eine Turing-Maschine  $M$ .

Gefragt: Terminiert  $M$  bei Eingabe von  $\langle M \rangle$ ?

Es gilt: Das spezielle Halteproblem ist unentscheidbar.

---

(Socher, S. 145)

---

**Satz 6.4**

---

Das allgemeine Halteproblem ist unentscheidbar.

---

(Socher, S. 146)

---

**Definition 6.6:** Spezielles Halteproblem mit leerem Eingabewort

---

Das *spezielle Halteproblem mit leerem Eingabewort* ( $H_\varepsilon$ ) lautet:

Gegeben: eine Turing-Maschine  $M$ .

Gefragt: Terminiert  $M$  bei Eingabe von  $\varepsilon$ ?

Es gilt: Das spezielle Halteproblem mit leerem Eingabewort ist unentscheidbar.

---

(Socher, S. 148)

---

**Definition und Satz 6.7:** Reduzierbarkeit

---

Die Sprache  $L_1 \subseteq \Sigma^*$  heißt *reduzierbar* auf die Sprache  $L_2 \subseteq \Gamma^*$ , falls es eine totale berechenbare Funktion  $f: \Sigma^* \rightarrow \Gamma^*$  gibt, sodass für alle  $w \in \Sigma^*$  gilt:

$w \in L_1$  genau dann wenn  $f(w) \in L_2$ .

Wir schreiben in diesem Fall  $L_1 \leq L_2$ . Es gilt:

Ist  $L_1 \leq L_2$  und ist  $L_2$  entscheidbar, so ist auch  $L_1$  entscheidbar, bzw. umgekehrt formuliert: Ist  $L_1$  unentscheidbar, so ist auch  $L_2$  unentscheidbar

---

(Socher, S. 149)

---



---

**Definition und Hilfssatz 6.8:** Modifiziertes Postisches Korrespondenzproblem

---

Das *modifizierte Postische Korrespondenzproblem* (MPKP) lautet:

Gegeben: Alphabete  $\Sigma$  und  $\Gamma$ , ein *Startzeichen*  $a \in \Sigma$  und zwei Funktionen  $\alpha, \beta : \Sigma \rightarrow \Gamma^*$

Gefragt: Gibt es ein Wort  $w \in \Sigma^*$  mit  $\alpha(aw) = \beta(aw)$ ?

Es gilt  $MPKP \leq PKP$ , das heißt, wenn das MPKP entscheidbar wäre, so wäre auch das PKP entscheidbar.

---

(Socher, S. 151)

**Errata S. 151:** Definition 6.8: Es gilt  $MPKP \leq PKP$ , das heißt wenn das MPKP unentscheidbar ist, so ist auch das PKP unentscheidbar.

**Errata S. 151:** In dem Beispiel zum MPKP muss in der rechten Tabelle die vorletzte Zeile so aussehen:

x            \*1\*0\*1\*1\*1\*            \*1\*0

---

---

**Satz 6.5**

---

Das PKP ist unentscheidbar.

---

(Socher, S. 152)

**Errata S. 152:** Das Wort  $\beta(u)$  ist also dem Wort  $\alpha(u)$  stets um genau einen Berechnungsschritt voraus.

**Errata S. 156:** Wir zeigen dies durch Reduktion des PKP auf das Schnittproblem.

---

---

**Satz 6.6**

---

Das Schnittproblem für kontextfreie Sprachen ist unentscheidbar.

---

(Socher, S. 156)

---

---

**Satz 6.7** Satz von Rice

---

Sei  $E$  irgendeine nichttriviale Eigenschaft berechenbarer Funktionen. Dann ist das folgende Problem unentscheidbar:

Gegeben: Eine Turing-Maschine  $M$ .

Gefragt: Hat  $f_M$  die Eigenschaft  $E$ ?

---

(Socher, S. 157)

**Errata S. 157:** es gibt mindestens eine Turing-Maschine  $Q$ , sodass  $f_Q$  die Eigenschaft  $E$  nicht besitzt.

## Hausaufgaben:

Bearbeiten sie bitte die Aufgabe 6.2

Sie können ihre Ergebnisse mit den Lösungsvorschlägen im Anhang abgleichen. Die Hausaufgaben werden nicht eingereicht.

## Mögliche Klausuraufgaben zu Kapitel 6:

- Multiple-Choice Questions zu Aussagen über Entscheidbarkeit (welche der folgenden Aussagen ist wahr? Bsp. Jede rekursive Sprache ist semi-entscheidbar.)
- Zeigen Sie mithilfe der Reduzierbarkeit, dass das spezielle Halteproblem mit leerem Eingabewort unentscheidbar ist.

## 7. Kapitel

### Definition 7.1: Die O-Notation

Seien  $f, g : \mathbb{N} \rightarrow \mathbb{R}$  zwei Funktionen. Wir sagen,  $f$  ist von der Ordnung  $g$ , falls es eine Konstante  $c > 0$  und eine natürliche Zahl  $n_0$  gibt, sodass gilt:

$$f(n) \leq c \cdot g(n) \text{ für alle } n \geq n_0.$$

Mit  $O(g)$  bezeichnen wir die Menge aller Funktionen der Ordnung  $g$ .

(Socher, S. 165)

### Satz 7.1

Sei  $f(n) = a_k n^k + a_{k-1} n^{k-1} + \dots + a_0$  ein Polynom vom Grad  $k$ . Dann gilt  $f = O(n^k)$ .

(Socher, S. 165)

### Tabelle 7.1: Die wichtigsten Komplexitätsklassen

Klasse	Ordnung
konstant	$O(1)$
logarithmisch	$O(\log n)$
linear	$O(n)$
$n$ -log- $n$	$O(n \log n)$
quadratisch	$O(n^2)$
polynomiell	$O(n^k)$ für $k \geq 1$
exponentiell	$O(d^n)$ für $d > 1$

(Socher, S. 166)

■ $f_1(n) = n$									■ $f_2(n) = n \cdot \log n$								
A	B	C	D	E	F	G	H		A	B	C	D	E	F	G	H	
1	2	3	4	5	6	7	8	1	0	2	3	8	10	12	14	24	1
9	10	11	12	13	14	15	16	2	27	30	33	36	39	42	45	64	2
17	18	19	20	21	22	23	24	3	68	72	76	80	84	88	92	96	3
25	26	27	28	29	30	31	32	4	100	104	108	112	116	120	124	160	4
33	34	35	36	37	38	39	40	5	165	170	175	180	185	190	195	200	5
41	42	43	44	45	46	47	48	6	205	210	215	220	225	230	235	240	6
49	50	51	52	53	54	55	56	7	245	250	255	260	265	270	275	280	7
57	58	59	60	61	62	63	64	8	285	290	295	300	305	310	315	384	8

Abbildung 7.8: Die Wachstumsfunktionen  $n$  und  $n \cdot \log n$  im Vergleich

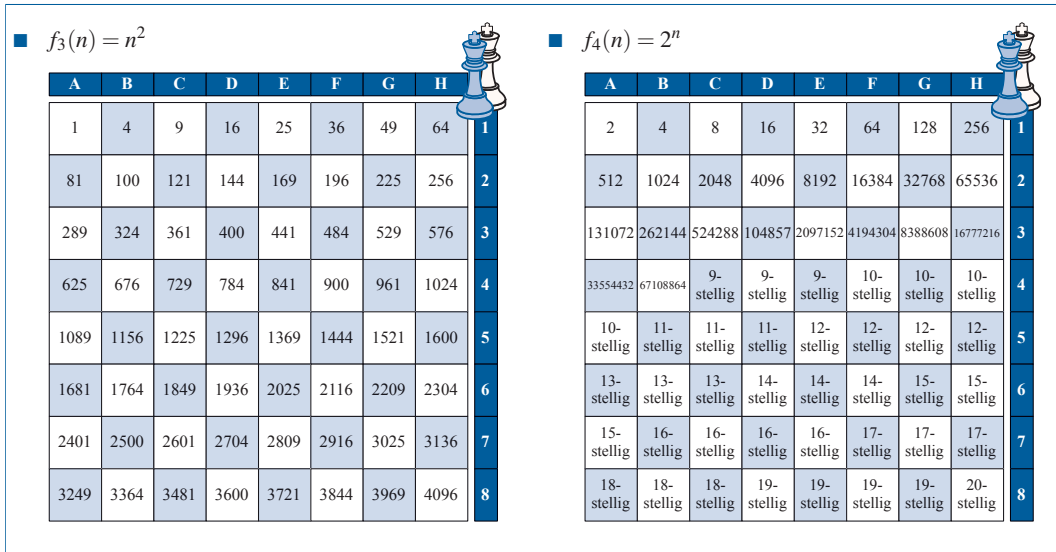


Abbildung 7.9: Die Wachstumsfunktionen  $n^2$  und  $2^n$  im Vergleich

(S. 357f aus Dirk W. Hoffmann, Theoretische Informatik, Hanser, 2. Auflage, 2011)

**Definition 7.2:** Laufzeitfunktion einer Turing-Maschine

Sei  $M$  eine deterministische oder nichtdeterministische (Mehrband)-Turing-Maschine. Die *Laufzeitfunktion*  $t_M: \Sigma^* \rightarrow \mathbb{N}$  ist folgendermaßen definiert:  $t_M(w) = n$ , falls  $n$  die minimale Länge einer terminierenden Berechnung  $(\#, z_0, w) \rightarrow^* (v, z_e, u)$  bei Eingabe von  $w$  ist. Hält  $M$  bei Eingabe von  $w$  nicht, so setzen wir  $t_M(w) = 0$ .

Die *Worst-case-Laufzeitfunktion*  $T_M: \mathbb{N} \rightarrow \mathbb{N}$  ist folgendermaßen definiert:

$$T_M(n) = \max(\{t_M(w) \mid w \in \Sigma^*, |w| = n\}).$$

(Socher, S. 169)

**Definition 7.3:** Die Klasse P

Eine Sprache  $L$  heißt *polynomiell*, falls es eine *deterministische* Turing-Maschine  $M$  mit  $L = \mathcal{L}(M)$  gibt, sodass

$$T_M = O(n^k) \text{ für ein } k \geq 0.$$

Die Klasse der polynomiellen Sprachen wird mit  $P$  bezeichnet.

(Socher, S. 169)

**Satz 7.2**

- a) Ist die Entscheidungsvariante des Rucksackproblems effizient lösbar, dann auch die Zahlvariante.
- b) Ist die Zahlvariante des Rucksackproblems effizient lösbar, dann auch die Optimierungsvariante.

(Socher, S. 171)

**Errata S. 176:** (in diesem Fall die Bepackung  $[0,0,0,0]$ )

**Errata S. 176:** (im Beispiel  $[0,0,0,0,1]$ )

**Errata S. 176:** In den Zeilen 8 und 9 des Listings muss G durch G1 ersetzt werden ( $\text{gewichte}(G1)$ ,  $\text{skalarprodukt}(X, G1, XG)$ )

. **Errata S. 176:** In den Zeilen 14 und 15 des Listings muss  $W$  durch  $W1$  ersetzt werden ( $\text{nutzen}(W1)$ ,  $\text{skalarprodukt}(X, W1, XW)$ )

---

**Definition 7.3:** Die Klasse  $P$

Eine Sprache  $L$  heißt *polynomiell*, falls es eine *deterministische* Turing-Maschine  $M$  mit  $L = \mathcal{L}(M)$  gibt, sodass

$$T_M = O(n^k) \text{ für ein } k \geq 0.$$

Die Klasse der polynomiellen Sprachen wird mit  $P$  bezeichnet.

(Socher, S. 177)

---

**Satz 7.3**

Die Entscheidungsvarianten der Probleme KP, CLIQUE, TSP, BPP und SAT sind in NP.

(Socher, S. 177)

**Errata S. 178:** die Erzeugung eines Lösungskandidaten (das Orakel):  $\text{bepackung}(X, N)$

**Errata S. 179:** Jede konkrete Instanz  $H$  des HP wird in diesem Beispiel in eine Instanz  $T$  des TSP transformiert,...

---

**Definition und Satz 7.5:** Polynomielle Reduzierbarkeit

Die Sprache  $L_1 \subseteq \Sigma^*$  heißt *polynomiell reduzierbar* auf die Sprache  $L_2 \subseteq \Gamma^*$ , falls es eine in polynomieller Zeit berechenbare Funktion  $f: \Sigma^* \rightarrow \Gamma^*$  gibt, sodass für alle  $w \in \Sigma^*$  gilt:

$$w \in L_1 \text{ gdw. } f(w) \in L_2.$$

Wir schreiben in diesem Fall  $L_1 \leq_p L_2$ . Es gilt:

Ist  $L_1 \leq_p L_2$  und ist  $L_2 \in P$  (bzw. NP), so ist auch  $L_1 \in P$  (bzw. NP).

(Socher, S. 179f)

---

**Definition 7.6:** NP-Vollständigkeit

Eine Sprache  $L$  heißt *NP-hart*, falls  $L' \leq_p L$  für jedes  $L' \in \text{NP}$  gilt.

Eine Sprache  $L$  heißt *NP-vollständig*, falls  $L \in \text{NP}$  und  $L$  NP-hart ist.

(Socher, S. 180)

---

**Satz 7.4**

Sei  $L$  NP-vollständig.

- Falls  $L \in P$ , so ist  $P = \text{NP}$ .
- Falls  $L \notin P$ , so ist  $L' \notin P$  für jedes NP-vollständige Problem  $L'$ .
- Ist  $L \leq_p L'$ , so ist  $L'$  NP-hart.

(Socher, S. 180)

**Satz 7.5** Satz von Cook

---

Das Erfüllbarkeitsproblem der Aussagenlogik (SAT) ist NP-vollständig.

---

(Socher, S. 181)

---

**Satz 7.6**

---

Das Problem 3-SAT ist NP-vollständig.

---

(Socher, S. 182)

**Errata S. 183:** Meiner Ansicht nach muss die folgende Bedingung gelten:  $I(w_i) = 1$ , wenn  $k \geq i + 2$ .

**Errata S. 183:** Dann lässt sich leicht nachrechnen, dass all Klauseln  $C_j$ ,  $j = 1, \dots, 4$  erfüllt sind.

**Errata S. 183:** Sei  $i \geq 1$  der kleinste Index, sodass ...

**Errata S. 183:** Ist  $I(w_i) = 1$  für alle  $i = 1, 2, 3$ , so ist ...

---

**Satz 7.7**

---

Das Problem CLIQUE ist NP-vollständig.

---

(Socher, S. 183)

---

**Satz 7.8**

---

Das Problem VC ist NP-vollständig.

---

(Socher, S. 184)

**Mögliche Klausuraufgaben zu Kapitel 6:**

- Multiple-Choice Questions zu Aussagen über Komplexität.
- Bestimmung der Komplexitätsklasse für sehr einfache Algorithmen.

## 8. Kapitel

---

**Definition 8.1:** Partition

---

Eine *Partition* einer Menge  $M$  ist eine Menge  $\{K_1, \dots, K_n\}$  mit  $K_i \subseteq M$  für  $i = 1, \dots, n$  mit folgender Eigenschaft: Für jedes  $x \in M$  gibt es genau ein  $K_i$  mit  $x \in K_i$ .

---

(Socher, S. 189)

---

**Definition 8.2:** Reflexivität, Symmetrie, Transitivität

---

Gegeben sei eine Relation  $R$  auf der Grundmenge  $M$ .

- a)  $R$  heißt *reflexiv*, falls  $x R x$  für alle  $x \in M$  gilt.
  - b)  $R$  heißt *symmetrisch*, falls für alle  $x, y \in M$  aus  $x R y$  stets  $y R x$  folgt.
  - c)  $R$  heißt *transitiv*, falls für alle  $x, y, z \in M$  aus  $x R y$  und  $y R z$  stets  $x R z$  folgt.
- 

(Socher, S. 190)

---

**Definition 8.3:** Äquivalenzrelation

---

Eine *Äquivalenzrelation* ist eine reflexive, symmetrische und transitive Relation.

---

(Socher, S. 190)

---

**Satz 8.1**

---

Seien  $M$  und  $N$  Mengen und sei  $f$  eine Abbildung von  $M$  in  $N$ . Dann ist die Relation  $\equiv_f$  auf  $M$ , die definiert ist durch:  $m_1 \equiv_f m_2$ , falls  $f(m_1) = f(m_2)$ , eine Äquivalenzrelation.

---

(Socher, S. 191)

---

**Satz 8.2**

---

Ist  $R$  eine Äquivalenzrelation auf der Menge  $M$ , so bildet die Menge der Äquivalenzklassen von  $R$  eine Partition von  $M$ .

---

(Socher, S. 191)

**Errata S. 191:** Offenbar bildet in Beispiel 8.5 a) und b) die Äquivalenzklassen eine Partition der ...

---

**Satz 8.3**

---

Sei  $\{K_1, \dots, K_n\}$  eine Partition von  $M$  und sei die Relation  $R$  auf  $M$  definiert durch  $xRy$ , falls sich  $x$  und  $y$  in der derselben Klasse befinden. Dann ist  $R$  eine Äquivalenzrelation.

---

(Socher, S. 192)

---

**Definition 8.4:** Graph

---

Ein Graph  $G = (V, E)$  besteht aus einer endlichen Menge  $V$ , deren Elemente Knoten (engl. vertices) genannt werden, und einer Menge  $E$  von Mengen  $\{u, v\}$  mit  $u, v \in V$  und  $u \neq v$ , die Kanten (engl. edges) genannt werden.

Ist  $e = \{u, v\}$  eine Kante zwischen  $u$  und  $v$ , so sagt man auch,  $e$  ist inzident mit  $u$  und  $v$  oder  $u$  und  $v$  sind adjazent (benachbart).

---

(Socher, S. 194)

**Errata S. 223:** Das Literaturverzeichnis ist eine Katastrophe und zeugt von einem fehlenden Lektorat.