

## versch. Sprachen & ihre Klassifizierungen

recap:  $L_{EA} = \{L(A) \mid A \text{ ist ein EA}\}$  ist die Menge der regulären Sprachen.

$L_{RE} = \{L(M) \mid M \text{ ist eine TM}\}$  ist die Menge aller rekursiv aufzählbaren Sprachen.

$L_R = \{L(M) \mid M \text{ ist eine TM, die immer hält}\}$  ist die Menge aller rekursiven Sprachen.

Algorithmus: Ein Programm, das auf jeder Eingabe  $x \in \Sigma^*$  hält.

$\text{Kod}(M) = h(\text{Code}(M))$  ist die Kodierung der TM  $M$  über  $\Sigma_{\text{bool}}$ .

$\text{Kod TM} = \{\text{Kod}(M) \mid M \text{ ist eine TM}\}$  ist die Menge der Kodierungen aller Turingmaschinen.

$$L_\emptyset := \{\text{Kod}(M) \mid L(M) = \emptyset\}$$

$$L_\emptyset^C := \{x \in (\Sigma_{\text{bool}})^* \mid x \neq \text{Kod}(\bar{M}) \quad \forall \text{TM } \bar{M} \quad \text{oder} \quad x = \text{Kod}(M) \quad \text{und} \quad L(M) \neq \emptyset\}$$

$$L_{\text{diag}} = \{w \in (\Sigma_{\text{bool}})^* \mid w = w_i \text{ für ein } i \in \mathbb{N}^* \text{ und } M_i \text{ akzeptiert } w; \text{ nicht}\}$$

$$L'_{\text{diag}} = \{w \in (\Sigma_{\text{bool}})^* \mid w = w_i \text{ für ein } i \in \mathbb{N}^* \text{ und } M_i \text{ akzeptiert } w \text{ nicht}\}$$

$$L_u = \{\text{Kod}(M) \# w \mid w \in (\Sigma_{\text{bool}})^* \text{ und } M \text{ akzeptiert } w\} = \text{universelle Sprache}$$

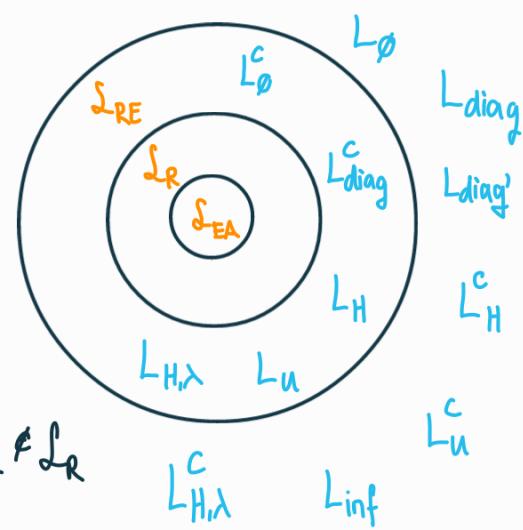
$$L_H = \{\text{Kod}(M) \# x \mid x \in (\Sigma_{\text{bool}})^* \text{ und } M \text{ hält auf } x\}$$

$$L_{H,\lambda} = \{\text{Kod}(M) \mid M \text{ hält auf } \lambda\}$$

$$L_{EQ} := \{\text{Kod}(M) \# \text{Kod}(\bar{M}) \mid L(M) = L(\bar{M})\}$$

$$\text{Kod}_L := \{\text{Kod}(M) \mid M \text{ ist eine TM und } L(M) = L\} \quad \text{Ricke} \Rightarrow \text{Kod}_L \notin L_R$$

$$L_{\text{inf}} := \{\text{Kod}(M) \mid M \text{ TM, die auf keinem Input hält}\}$$



## 5.3 Die Methode der Reduktion

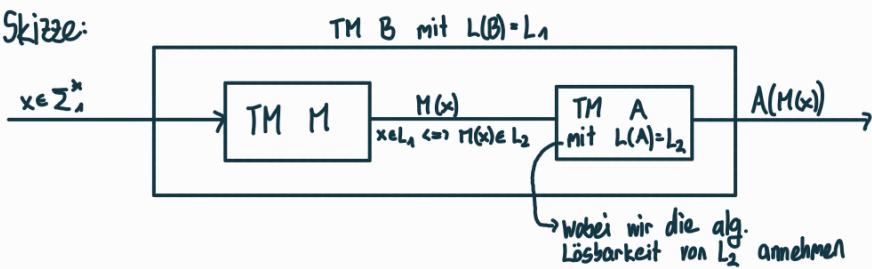
Def: Seien  $L_1 \subseteq \Sigma_1^*$ ,  $L_2 \subseteq \Sigma_2^*$  zwei Sprachen.

$L_1$  ist auf  $L_2$  reduzierbar,  $L_1 \leq_R L_2$ ,

falls  $L_2 \in \mathcal{L}_R \Rightarrow L_1 \in \mathcal{L}_R$ .

Intuitiv:  $L_2$  ist bzgl. der algorithmischen Lösbarkeit mindestens so schwer wie  $L_1$ .

Skizze:



Def: Seien  $L_1 \subseteq \Sigma_1^*$  und  $L_2 \subseteq \Sigma_2^*$  zwei Sprachen.

$L_1$  ist EE-reduzierbar auf  $L_2$ ,  $L_1 \leq_{EE} L_2$ ,

falls eine TM M existiert, die eine Abb.  $f_M: \Sigma_1^* \rightarrow \Sigma_2^*$  berechnet,

s.d.  $\forall x \in \Sigma_1^*: x \in L_1 \iff f_M(x) \in L_2$ .

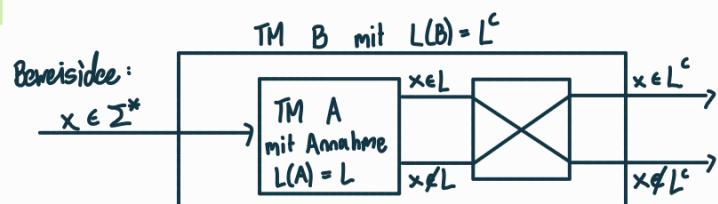
Wir sagen dann, dass die TM M die Sprache  $L_1$  auf die Sprache  $L_2$  reduziert.

Lemma 5.3: Seien  $L_1 \subseteq \Sigma_1^*$  und  $L_2 \subseteq \Sigma_2^*$  zwei Sprachen.

Dann gilt:  $L_1 \leq_{EE} L_2 \Rightarrow L_1 \leq_R L_2$

Lemma 5.4: Sei  $\Sigma$  ein Alphabet.

$\forall L \subseteq \Sigma^*: L \leq_R L^c \text{ und } L^c \leq_R L$



Bspk: Warum ist  $L_H \in \mathcal{L}_{RE} \setminus \mathcal{L}_R \wedge L_H^c \notin \mathcal{L}_{RE}$  kein Widerspruch zu  $L_H^c \leq_R L_H$ ?

Die  $\leq_R$ -Reduktion setzt die Annahme  $L_H \in \mathcal{L}_R$  voraus, was in diesem Fall nicht tatsächlich erfüllt ist.

Somit sagt  $L_1 \leq_R L_2$  nichts darüber aus, in welchem Verhältnis  $L_1$  und  $L_2$  außerhalb von  $\mathcal{L}_R$  sind.

Bmk: Das folgende Lemma steht nicht im Skript, wir beweisen es aber hier in der Übungsstunde und ihr dürft es in Beweisen verwenden.

**Lemma 5:**  $(L \in \mathcal{L}_{RE} \wedge L^c \in \mathcal{L}_{RE}) \Rightarrow L \in \mathcal{L}_R$  (und somit auch  $L^c \in \mathcal{L}_R$ )

**Beweis:**

- Seien  $M, M'$  TM mit  $L(M) = L$  und  $L(M') = L^c$ .
- Wir konstruieren den Algo A wie folgt:
  - Simuliere abwechselnd je einen Schritt von M und M' auf x.
  - Sobald eine der beiden Simulationen hält, entscheidet A wie folgt:
    - Falls M akz.  $\rightarrow$  akz.
    - Falls M verw.  $\rightarrow$  verw.
    - Falls  $M'$  akz.  $\rightarrow$  verw.
    - Falls  $M'$  verw.  $\rightarrow$  akz.
- Da  $x \in L \vee x \in L^c$  gilt, wird M oder  $M'$  halten und x akz. (oder verw.)  
Also hält A auf jeder Eingabe  $x \in \Sigma^*$ .  $\square$

Bmk: Anwendung der Reduktionsmethoden:

1) Ziel:  $Z_1: L \notin \mathcal{L}_R$

Vorgehen: a) Wir wählen ein passendes  $L' \notin \mathcal{L}_R$  und zeigen  $L' \leq_R L$ .  
b) Satz von Rice (nächste Woche)

2) Ziel:  $Z_2: L \notin \mathcal{L}_{RE} \quad L \notin \mathcal{L}_R \quad L^c \in \mathcal{L}_{RE}$

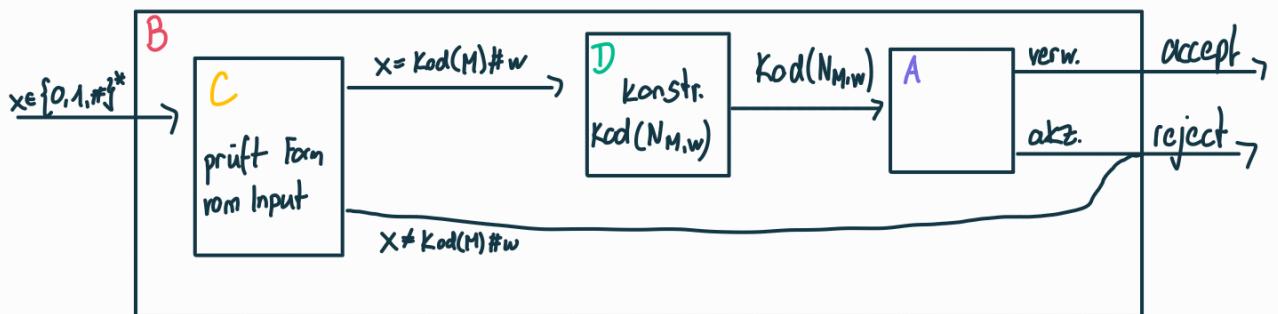
Vorgehen: a) Wir wählen ein passendes  $L' \notin \mathcal{L}_{RE}$  und zeigen  $L' \leq_{EE} L$ .  
(wird im worked example behandelt)

b) Wir zeigen  $L^c \in \mathcal{L}_{RE}$  und  $L \notin \mathcal{L}_R$

Aufg 1) Sei  $L_{\text{inf}} := \{\text{Kod}(M) \mid M \text{ TM, die auf keinem Input hält}\}$   
 $\exists_2: L_{\text{inf}} \notin \mathcal{L}_2$

Lösung:

- Wir zeigen  $L_H \leq_R L_{\text{inf}}$  (denn  $L_{\text{inf}} \in \mathcal{L}_2 \Rightarrow L_H \in \mathcal{L}_2 \not\models$ )
- Angenommen  $L_{\text{inf}} \in \mathcal{L}_2$ , also  $\exists A$  Algo mit  $L(A) = L_{\text{inf}}$ .
- Wir konstruieren einen Algo  $B$  mit  $L(B) = L_H$



B tut folgendes:

C | • bekommt Input  $x \in \{0,1,\#\}^*$

| • prüft, ob  $x = \text{Kod}(M) \# w$

| - falls nein  $\Rightarrow$  reject

| - falls ja: | • konstruiere  $\text{Kod}(N_{M,w})$

D | wo bei  $N_{M,w}$  eine TM ist, die ihren Input ignoriert und M auf w simuliert.

| • Gebe  $\text{Kod}(N_{M,w})$  als Eingabe an A

A | - falls A akz.  $\Rightarrow$  reject

| - falls A verw.  $\Rightarrow$  accept

Offensichtlich hält B.

• Korrektheitsbeweis:  $Z_2: L(B) = L_H$

" $\supseteq$ " Sei  $x \in L_H$ .  
 $\Rightarrow x = \text{Kod}(M) \# w$  und  $M$  hält auf  $w$   
 $\Rightarrow N_{M,w}$  hält auf jedem Input  
 $\Rightarrow \text{Kod}(N_{M,w}) \notin L_{\text{inf}}$   
 $\Rightarrow A$  verwirft  $\text{Kod}(N_{M,w})$   
 $\Rightarrow B$  akzeptiert  
 $\Rightarrow x \in L(B)$

" $\subseteq$ " Sei  $x \in L(B)$   
 $\Rightarrow x = \text{Kod}(M) \# w$  und  $A$  verwirft  $\text{Kod}(N_{M,w})$   
 $\Rightarrow N_{M,w}$  hält auf mind. einem Input  
 $\Rightarrow N_{M,w}$  hält immer  
 $\Rightarrow M$  hält auf  $w$   
 $\Rightarrow \text{Kod}(M) \# w = x \in L_H$   $\square$

Aufg 2) Sei  $L_U := \{\text{Kod}(M) \# w \mid w \in (\Sigma_{\text{sol}})^* \text{ und } M \text{ akzeptiert } w\}$  (universelle Sprache)

Sei  $L_{\text{union}} := \{\text{Kod}(M) \# \text{Kod}(M') \# w : w \in L(M) \cup L(M')\}$

$Z_2: L_U \leq_{EE} L_{\text{union}}$

Lösung: • Wir beschreiben eine TM  $M$ , die eine Abb.  $f_M: \{0, 1, \#\} \rightarrow \{0, 1, \#\}$  berechnet, s.d.  $x \in L_U \Leftrightarrow f_M(x) \in L_{\text{union}}$

•  $M$  arbeitet wie folgt:

Für eine Eingabe  $w$  überprüft  $M$ ,

ob  $w$  die Form  $w = x \# y$  für  $x, y \in \{0, 1\}^*$  hat.

- falls nein, ist  $f_M(w) = \lambda$ , da  $\lambda \in L_{\text{union}}$  und  $w \notin L_U$ .

- falls ja, hält  $M$  mit dem Inhalt  $x \# x \# y$  auf dem Band.

Korrektheitsbeweis: • Für  $w = x \# y$ ,  $x, y \in \{0, 1\}^*$  gilt:

$w = x \# y \in L_U \Leftrightarrow x = \text{Kod}(A)$  wobei  $y \in L(A)$   
 $\Leftrightarrow \text{Kod}(A) \# \text{Kod}(A) \# y \in L_{\text{union}}$   
 $\Leftrightarrow x \# x \# y \in L_{\text{union}}$   
 $\Leftrightarrow f_M(w) \in L_{\text{union}}$

• Ansonsten gilt  $w \notin L_U \Leftrightarrow f_M(w) = \lambda \notin L_{\text{union}}$   $\square$