# Beweismuster für TI

## Beweise der Form $L \in \mathcal{L}_{\text{RE}}$ Eingabe der Form Kod $(M)$

Hier geht es darum, zu zeigen, dass es eine TM gibt, welche für alle Worte in der Sprache $L$ **eine** akzeptierende Berechnung aufweist. Dies wird immer gleich gemacht. Wir machen die folgenden Schritte:

1. Konstruiere eine NTM $N$ welche zuerst **deterministisch** prüft, ob der Input der Form $w = \text{Kod}(M)$ für eine TM $M$ ist. Falls $w$ keine TM kodiert akzeptiere / verwerfe, je nach dem wie die Sprache definiert ist.
2. Falls $x = \text{Kod}(M)$ für eine TM M, dann wählt $N$ **nichtdeterministisch** ein Wort $x \in \{0,1\}^*$ und simuliert **deterministisch** die Berechnung von $M$ auf $x$.
3. Wir können hier nur ein Statement machen, falls $M$ das Wort $x$ akzeptiert, da nicht-akzeptierende Berechnungen nicht halten müssen. Es ist wichtig, dass wir eine Aussage machen, fall das gewählte Wort akzeptiert wird und dann sagen, dass falls kein Wort existiert, welches akzeptiert wird, dass die Berechnung unendlich lange geht.

Das ganze illustriert an zwei Beispielen:

### HS16 Aufgabe 1 a

$$L_{\text{nohalt}} = \{\text{Kod}(M) \mid M \text{ ist eine TM , die auf keiner Eingabe hält.}\}$$

**Zeigen Sie, dass** $(L_{\text{nohalt}})^{\complement} \in \mathcal{L}_{\text{RE}}$ Wir betrachten zuerst die Sprache

$$\begin{aligned}(L_{\text{nohalt}})^{\complement} = \quad & \{w \in \{0,1\} \mid w \neq \text{Kod}(M) \text{ für alle TM } M\} \\ & \cup \{\text{Kod}(M) \mid \text{ es gibt eine Eingabe } x, \text{ so dass } M \text{ auf } x \text{ hält}\}\end{aligned}$$

Wir bechreiben eine NTM $N$ mit $L(N) = (L_{\text{nohalt}})^{\complement}$. Es gibt somit für jedes Wort $w \in (L_{\text{nohalt}})^{\complement}$ eine **endliche**, **akzeptierende** Berechnung von $N$ auf $w$. $N$ überprüft zuerst deterministisch, ob die $w = \text{Kod}(M)$ für irgendeine TM $M$. Ist dies nicht der Fall, dann akzeptiert $N$ die Eingabe $w$. Ansonsten wählt $N$ nichtdeterministisch ein Wort $x$ über dem Eingabealphabet von $M$ und simuliert $M$ deterministisch auf $x$.
Falls $M$ auf dem Wort $x$ hält, dann akzeptiert $N$ die Eingabe $w$. Falls $M$ auf $x$ unendlich lange rechnet, dann ist auch die Berechnung von $N$ auf $w$ unendlich. $N$ akzeptiert alle Wörter, welche nicht die Kodierung einer TM sind. Falls die Eingabe $w$ eine Kodierung einer TM $M$ ist, dann ist $w \in (L_{\text{nohalt}})^{\complement}$ genau dann, wenn es ein Wort $x$ im Eingabealphabet gibt, so dass $M$ auf $x$ hält. Es gibt somit eine akzeptierende Berchnung von $N$ auf $w$, in der $N$ genau dieses Wort $x$ nichtdeterministisch wählt. Falls es keine akzeptierende Berechnung von $N$ auf $w$ gibt, dann gilt $w \in L_{\text{nohalt}}$. Somit ist $N$ eine NTM , welche $(L_{\text{nohalt}})^{\complement}$ akzeptiert, damit gilt per Definition $(L_{\text{nohalt}})^{\complement} \in \mathcal{L}_{\text{RE}}$

**HS17 Aufgabe 1 b**

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

**Zeigen Sie, dass** $(L_{\text{empty}})^{\complement} \in \mathcal{L}_{\text{RE}}$ Wir betrachten zuerst die Sprache

$$\begin{aligned}(L_{\text{empty}})^{\complement} = \quad & \{w \in \{0,1\} \mid w \neq \text{Kod}(M) \text{ für alle TM } M\} \\ & \cup \{\text{Kod}(M) \mid \text{ es gibt eine Eingabe } x, \text{ so dass } M \text{ } x \text{ akzeptiert}\}\end{aligned}$$

Wir konstruieren eine NTM $N$, für welche $L(N) = (L_{\text{empty}})^{\complement}$ gilt. $N$ prüft zuerst, ob die Eingabe $w$ eine TM $M$ kodiert ($w = \text{Kod}(M)$ für eine TM $M$), falls dem nicht so ist, so akzeptiert $N$ die Eingabe $w$. Ansonsten wählt $N$ nichtdeterministisch ein Wort $x$ über dem Eingabealphabet von $M$ und simuliert deterministisch die Arbeit von $M$ auf $x$. Falls $M$ das Wort $x$ akzeptiert, so akzeptiert $N$ die Eingabe $w$. Falls $M$ das Wort $x$ verwirft, so akzeptiert $N$ in diesem Lauf die Eingabe $w$ nicht. Falls die Berechnung von $M$ auf $x$ unendlich ist, so rechnet $N$ in diesem Lauf auf $w$ unendlich lange und akzeptiert das Wort $w$ in diesem Lauf nicht.
$N$ akzeptiert alle Wörter, welche keine TM kodieren und akzeptiert $w = \text{Kod}(M)$ für eine TM $M$ genau dann, wenn es ein Wort $x$ über dem Eingabealphabet von $M$ gibt und eine akzeptierende Berechnung von $N$ auf $w$ gibt, in welcher $M$ das Wort $x$ akzeptiert. Falls keine solche Berechnung für jedes Wort existiert, dann ist $w \in L_{\text{empty}}$. Somit sehen wir, dass $L(N) = (L_{\text{empty}})^{\complement}$.

## Beweise der Form $L \notin \mathcal{L}_{\mathrm{R}}$

Hier wollen wir eine Sprache $L'$ wählen, für welche wir bewiesen haben, dass $L' \notin \mathcal{L}_{\mathrm{R}}$ gilt. Die folgenden Sprachen, haben Sätze / Lemmas / Korollare im Skript und verweise darauf sollten ohne Beweis erfolgen dürfen.

- $L_{\mathrm{U}} \notin \mathcal{L}_{\mathrm{R}}$ (Satz 5.7.)
- $L_{\mathrm{H}} \notin \mathcal{L}_{\mathrm{R}}$ (Satz 5.8.)
- $L_{\mathrm{empty}} \notin \mathcal{L}_{\mathrm{R}}$ (Korollar 5.4.)
- $L_{\mathrm{H},\lambda} \notin \mathcal{L}_{\mathrm{R}}$ (Lemma 5.8.)

Wir beachten dabei, dass das folgende Lemma (5.4.) gilt für eine Sprache $L$

$$L \leq_{\mathrm{R}} L^{\mathtt{C}} \text{ und } L^{\mathtt{C}} \leq_{\mathrm{R}} L$$

Nach dem wir eine Sprache gewählt haben, machen wir das folgende. Wir nehmen an, dass $L \in \mathcal{L}_{\mathrm{R}}$ und konstruieren dann eine R-Reduktion der Form $L' \leq_{\mathrm{R}} L$, wobei $L' \notin \mathcal{L}_{\mathrm{R}}$ gilt. Damit erhalten wir einen Widerspruch und es folgt $L \notin \mathcal{L}_{\mathrm{R}}$. Die R-Reduktion darf wegen der Annahme, dass $L \in \mathcal{L}_{\mathrm{R}}$ gilt einen Algorithmus $A$ annehmen der die Sprache $L$ erkennt, dieser wird dann als Subprogramm für einen Algorithmus verwendet, welcher $L'$ erkennt. Dies können wir an Hand von einem Beispiel illustrieren.

### HS17 Aufgabe 1 a

$L_{\cap=\emptyset} = \{\mathrm{Kod}(M_1) \# \mathrm{Kod}(M_2) \mid M_1 \text{ und } M_2 \text{ sind TM und } L(M_1) \cap L(M_2) = \emptyset\}$

**Zeigen Sie, dass** $L_{\cap=\emptyset}$ Wir wählen $L_{\mathrm{empty}}$ und zeigen die R-Reduktion $L_{\mathrm{empty}} \leq_{\mathrm{R}} L_{\cap=\emptyset}$. Sei $A$ ein Algorithmus mit $L(A) = L_{\mathrm{empty}}$. Wir konstruieren einen Algorithmus $B$ folgendermassen, $B$ prüft zuerst ob die Eingabe $w$ die Kodierung einer TM $M$ ist. Ist dies nicht der Fall, dann verwirft $B$ die Eingabe $w$. Ist $w = \mathrm{Kod}(M)$ für eine TM $M$, dann konstruiert $B$ die Eingabe $w' = \mathrm{Kod}(M) \# \mathrm{Kod}(M)$. Diese Eingabe wird dann an den Algorithmus $A$ weitergegeben. Gilt $w' \in L_{\cap=\emptyset}$, dann muss $L(M) = \emptyset$ gelten und somit $w \in L_{\mathrm{empty}}$, gilt $L(M) \notin \emptyset$, dann muss $L(M) \neq \emptyset$ gelten und somit auch $w \notin L_{\mathrm{empty}}$. Somit kann $B$ die Arbeit von $A$ auf $w'$ simulieren und die Ausgabe direkt übernehmen. Damit haben wir die R-Reduktion $L_{\mathrm{empty}} \leq_{\mathrm{R}} L_{\cap=\emptyset}$ gezeigt und wir wissen, das $L_{\mathrm{empty}} \notin \mathcal{L}_{\mathrm{R}}$ gilt aus der Vorlesung. Somit kann $L_{\cap=\emptyset} \in \mathcal{L}_{\mathrm{R}}$ nicht gelten und wir haben die Aussage gezeigt.

## Pumping-Lemma für kontextfreie Sprachen

**Lemma 10.6 (Pumping-Lemma für kontextfreie Sprachen).** Sei $L$ kontextfrei. Dann existiert eine nur von $L$ abhängige Konstante $n_L$, so dass für alle Wörter $z \in L$ mit $|z| \geq n_L$ eine Zerlegung

$$z = uvwxy$$

von $z$ existiert, so dass

$$\begin{aligned} &(i)\ |vx| \geq 1, \\ &(ii)\ |vxx| \leq n_L \text{ und} \\ &(iii)\ \left\{uv^i wx^i y \mid i \in \mathbb{N}\right\} \subseteq L. \end{aligned}$$

Wir zeigen nun an Hand eines Beispiels, wie man mit Hilfe des Pumping-Lemmas zeigen kann, dass eine Sprache nicht kontextfrei ist. Sei die Sprache

$$L = \left\{a^i b^i c^i \mid i, j, k \in \mathbb{N} \text{ und } k = \min\{i, j\}\right\}$$

gegeben, wir zeigen nun, dass diese nicht kontextfrei ist. Nehmen wir an $L$ ist kontextfrei ($L \in \mathcal{L}_2$), dann sei $n_L$ gemäss dem Pumping-Lemma, wir betrachten $z := a^{n_L} b^{n_L} c^{n_L} \in L$ und wir betrachten die Worte $z_i := uv^i wx^i y$ für die Zerlegung von $z$ gemäss dem Pumping Lemma, welche existiert, da $|z| \geq n_L$. Es gelte per Pumping-Lemma

$$(i)\ |vx| \geq 1, \quad (ii)\ |vwx| \leq n_L$$

Wir betrachten nun die folgenden fünf möglichen Fälle, welche alle möglichen Zerlegungen abdecken. Wir betrachten dabei welche Symbole aus $\{a, b, c\}$ $v$ und $x$ enthalten, wobei wir (i) beachten:

1. $a$ und $b$ : Wir pumpen so lange, bis $|z|_a > n_L$ und $|z|_b > n_L$ gilt und somit $k = n_L < \min\{|z|_a, |z|_b\}$ und somit das P.-Lemma verletzt ist, das das gepumpte Wort nicht in $L$ ist.

2. $b$ und $c$ : Wir pumpen so lange bis $|z|_c > n_L$ und $|z|_b > n_L$ und somit $\min\{|z|_a, |z|_b\} = n_L < k$ und somit wiederum das P.-Lemma verletzt ist.

3. $a$ : Nach (i) gilt, dass $|vx| \geq 1$ und somit gilt für $z_0$, dass $|z_0|_a = i' < n_L$, da mindestens ein $a$ wegfällt und somit ist $i' < j$ und wegen $i' < n_L = k$, haben wir einen Widerspruch zum P.-Lemma.

4. $b$ : Dieser Fall folgt analog zum Fall für nur $a$.

5. $c$ : Für $z_2$ gilt wegen (i) $|z_2| > n_L$ und somit gilt $k > \min\{i, j\}$, wiederum ein Widerspruch zum P.-Lemma.

Andere Fälle sind wegen (i) und (ii) ausgeschlossen und somit ergibt sich in jedem Fall ein Widerspruch zum P.-Lemma und wir haben $L \in \mathcal{L}_\in$ gezeigt.

## $L \notin \mathcal{L}_{\mathrm{RE}}$ mit Satz von Rice

Diese Beweise enthalten fast immer den Satz "Sie dürfen hierfür alle aus der Vorlesung bekannten Resultate verwenden." oder zumindest was ähnliches. Der Satz von Rice ist gegeben durch.

**Satz von Rice (5.9)** Jedes semantisch nichttriviale Entscheidungsproblem über Turingmaschinen ist unentscheidbar. Ein solches Problem muss die folgenden Eigenschaften erfüllen:

1. (i) Es gibt eine TM $M_1$, so dass $\mathrm{Kod}(M_1) \in L$ (also $L \neq \emptyset$)
2. (ii) Es gibt eine TM $M_2$, so dass $\mathrm{Kod}(M_2) \notin L$ (also $L \neq \mathrm{KodTM}$)
3. (iii) für zwei Turingmaschinen $A$ und $B$ impliziert $L(A) = L(B)$

$$\mathrm{Kod}(A) \in L \iff \mathrm{Kod}(B) \in L$$

Die Eigenschaft (iii) ist meistens die wichtigste zu prüfen, also prüfe diese zuerst, falls nicht klar ist ob Satz von Rice anwendbar ist. Wir beweisen dann im ersten Schritt, dass die gegebene Sprache nicht in $\mathcal{L}_{\mathrm{R}}$ ist nach dem Satz von Rice und betrachten das Komplement, für welches wir beweisen, dass es in $\mathcal{L}_{\mathrm{RE}}$ ist. Wäre dann die Sprache selbst auch in $\mathcal{L}_{\mathrm{RE}}$, dann würde die Sprache in $\mathcal{L}_{\mathrm{R}}$ sein. Dies folgt daraus, dass wir eine TM konstruieren kann, welche TM für $L$ und $L^{\complement}$ auf einer Eingabe simuliert und dann eine davon akzeptieren muss in endlicher Zeit und wir dadurch eine TM konstruiert haben, welche die Sprache erkennt. Ein Beispiel dazu

**HS21 Aufgabe 2**

$$L_{111} = \{\mathrm{Kod}(M) \mid 111 \in L(M)\}$$

**Zeigen Sie, dass** $L_{111}^{\complement} \notin \mathcal{L}_{\mathrm{RE}}$ gilt. Sie dürfen hierfür alle aus der Vorlesung bekannten Resultate verwenden.
Wir zeigen zuerst, dass $L_{111}$ ein semantisch nichttriviales Entscheidungsproblem über Turingmaschinen ist. Das dies ein Entscheidungproblem über Turingmaschinen ist, ist klar. Die TM , welche nur 111 akzeptiert ist in $L_{111}$ enthalten und die TM , welche alle Worte verwirft ist nicht in $L_{111}$ enthalten. Somit sind die Bedingungen (i) und (ii) im Satz von Rice erfüllt. Seien $A$ und $B$ zwei beliebige TM . Falls $L(A) = L(B)$, dann akzeptieren entweder beide TM das Wort 111 oder beide verwerfen das Wort 111. Somit gilt $\mathrm{Kod}(A) \in L \iff \mathrm{Kod}(B) \in L$. Damit folgt aus dem Satz von Rice, dass $L_{111} \notin \mathcal{L}_{\mathrm{R}}$. Wir zeigen nun, dass $L_{111} \in \mathcal{L}_{\mathrm{R}}$, dies gilt, da wir eine TM $M'$ angeben, welche die Eingabe prüft, ist die Eingabe nicht von der Form $\mathrm{Kod}(M)$ für eine TM $M$, dann verwirft $M'$ direkt, ansonsten simuliert $M'$ die Arbeit von $M$ auf 111 und akzeptiert genau dann, wenn $M$ die Eingabe 111 akzeptiert.
Nehmen wir nun $L_{111}^{\complement} \in \mathcal{L}_{\mathrm{RE}}$ an, so haben wir wegen dem obigen Argumen (das müsste man dazu schreiben) einen Widerspruch und somit folgt $L_{111}^{\complement} \notin \mathcal{L}_{\mathrm{RE}}$.

## Typische Beweismuster

Bei R-Reduktionen oder EE-Reduktionen gibt es folgende immer wiedervorkommende Modifikationen von Turingmaschinen:

### Beweis $L_{\mathrm{H}} \leq_{\mathrm{R}} L_{\mathrm{U}}$

Wir nehmen an, es gelte $L_{\mathrm{H}} \in \mathcal{L}_{\mathrm{R}}$, dann gibt es einen Algorithmus $A$, welcher $L_{\mathrm{H}}$ entscheidet. Sei nun eine Eingabe $w$ für welche wir prüfen wollen, ob $w \in L_{\mathrm{U}}$ oder nicht. Dafür konstruieren wir einen Algorithmus $B$, welcher zuerst prüft, ob $w$ die Form $\mathrm{Kod}\,(M)\,\#x$ hat, falls nicht, dann verwirft $B$ sofort. Falls $w = \mathrm{Kod}\,(M)\,\#x$ gilt, dann berechnet $B$ eine Eingabe $w' = \mathrm{Kod}\,(M')\,\#x$, wobei $M'$ die TM ist, welche man erhält, wenn man alle Zustandswechsel in $M$ zu $q_{\mathrm{reject}}$ in eine Endlosschlaufe umleitet. $B$ simuliert dann die Arbeit von $A$ auf $w'$, wobei $M'$ genau dann auf $x$ hält, wenn $M$ die Eingabe $x$ akzeptiert. Somit gilt nun

$$\begin{aligned} w' \in L_{\mathrm{H}} &\iff M' \text{ hält auf } x \\ &\iff M \text{ akzeptiert } x \\ &\iff w \in L_{\mathrm{U}} \end{aligned}$$

Daraus folgt, dass $L_{\mathrm{H}} \leq_{\mathrm{R}} L_{\mathrm{U}}$. Wäre der Beweis $L_{\mathrm{H}} \leq_{\mathrm{EE}} L_{\mathrm{U}}$ gefragt, so würden wir statt zu verwerfen, falls $w$ nicht von der Form $\mathrm{Kod}\,(M)\,\#x$ wäre, die Eingabe zu $\mathrm{Kod}\,(M')\,\#x$ ändern, wobei $M'$ ihren Input ignoriert und in einen Endlosschlaufe geht, also nie hält, wobei $x = \lambda$.

### Beweis $L_{\mathrm{U}} \leq_{\mathrm{R}} L_{\mathrm{H}}$

Wir nehmen an, es gelte $L_{\mathrm{U}} \in \mathcal{L}_{\mathrm{R}}$, dann gibt es einen Algorithmus $A$, welcher $L_{\mathrm{U}}$ entscheidet. Sei nun eine Eingabe $w$ für welche wir prüfen wollen, ob $w \in L_{\mathrm{H}}$ oder nicht. Dafür konstruieren wir einen Algorithmus $B$, welcher zuerst prüft, ob $w$ die Form $\mathrm{Kod}\,(M)\,\#x$ hat, falls nicht, dann verwirft $B$ sofort. Falls $w = \mathrm{Kod}\,(M)\,\#x$ gilt, dann berechnet $B$ eine Eingabe $w' = \mathrm{Kod}\,(M')\,\#x$, wobei $M'$ die TM ist, welche man erhält, wenn man alle Zustandswechsel in $M$ von $q_{reject}$ zu $q_{accept}$ umleitet. $B$ simuliert dann die Arbeit von $A$ auf $w'$, wobei $M'$ genau dann $x$ akzeptiert, falls $M$ auf $x$ hält. Somit gilt nun

$$\begin{aligned} w' \in L_{\mathrm{U}} &\iff M' \text{ akzeptiert } x \\ &\iff M \text{ hält auf } x \\ &\iff w \in L_{\mathrm{H}} \end{aligned}$$

Daraus folgt, dass $L_{\mathrm{U}} \leq_{\mathrm{R}} L_{\mathrm{U}}$. Wäre der Beweis $L_{\mathrm{H}} \leq_{\mathrm{EE}} L_{\mathrm{U}}$ gefragt, so würden wir statt zu verwerfen, falls $w$ nicht von der Form $\mathrm{Kod}\,(M)\,\#x$ wäre, die Eingabe zu $\mathrm{Kod}\,(M')\,\#x$ ändern, wobei $M'$ ihren Input ignoriert und direkt in $q_{reject}$ übergeht (also alle Eingaben verwirft) und $x = \lambda$ ist.

## Diverses

- Aus der Aussage $L_1 \leq_{\mathrm{R}} L_2$ lässt sich nur eine Aussage über $\mathcal{L}_{\mathrm{R}}$ machen, keine Aussagen über $\mathcal{L}_{\mathrm{RE}}$. Es kann sein, dass $L_1 \notin \mathcal{L}_{\mathrm{RE}}$ und $L_1 \leq_{\mathrm{R}} L_2$ gilt, aber es gilt $L_1 \in \mathcal{L}_{\mathrm{RE}}$. Eine Aussage lässt sich nur über eine EE-Reduktion machen. Dies lässt sich damit begründen, dass die Annahme der R-Reduktion, dass es für $L_1$ eine TM gibt, welche die Sprache entscheidet, bereits ein falsche Aussage ist und somit kann man alle Aussagen daraus folgern.