TI Übungsstunde 8

Marcel Schmid marcesch@student.ethz.ch

4.11.2020

1 Alte Serie und allgemeine Infos

- Potenzmengenkonstruktion: kurze Erklärung, was ihr da genau macht, nicht vergessen – nicht einfach nur Diagramm/Tabelle hinhauen
- Einzelne Punkte immer noch nicht "wichtig", i.e. es wird keine Bonusnote geben aus den Punkten (immernoch 50% Grenze)

2 Theorie

2.1 Reduktionen

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

  $\Rightarrow$ Wie zeigen wir das? Wir nehmen an, dass wir eine Subroutine A haben für L_2. Wir erhalten eine Eingabe w für L_1

  $\Rightarrow$ Konstruiere aus w eine Eingabe w_A für Subroutine A. Sei $A(w_A)$ das Resultat der Berechnung von A

  $\Rightarrow$ Wir wollen dann aus $A(w_A)$ unsere definitive Lösung berechnen (meistens eins zu eins übernehmen oder allenfalls invertieren).

  $\Rightarrow$ Die $\leq_R$ Reduktion ist meistens intuitiver und weniger formal, aber auch weniger mächtig (da $\leq_{EE} \Rightarrow \leq_R$)

- $L_1 \leq_{EE} L_2$ falls es eine TM M gibt, welche eine Funktion f_M berechnet so dass $x \in L_1 \iff f_M(x) \in L_2$ gilt.

  $\Rightarrow$ Etwas schwieriger (unintuitiver) zu zeigen, aber "sicherer" $\Rightarrow$ es können weniger (gravierende) Fehler passieren.

  $\Rightarrow$ Und man zeigt es fast nie verkehrt herum...

  $\Rightarrow$ Für **alle** EE-Reduktionsbeweise muss man die Doppelimplikation zeigen (daher ist die EE-Reduktion auch sehr "Fail-Safe", da ihr an dem Beweis scheitert, falls ihr etwas falsch macht)

2.2 $L, L^C \in \mathcal{L}_{RE} \Rightarrow L \in \mathcal{L}_R$

Sehr nützlicher Trick um gewisse Dinge zu zeigen (Achtung: wenn nicht in der VL gezeigt, dann müsstet ihr das nochmals in explizit zeigen):

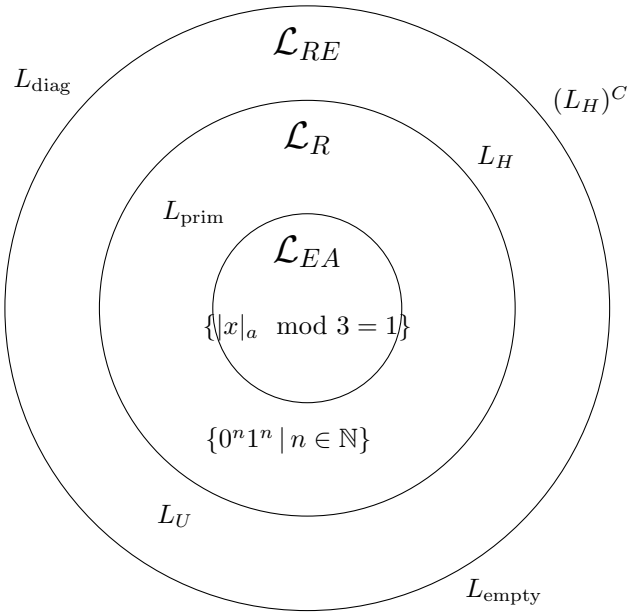
Sei L eine Sprache über Σ. Falls sowohl L und L^C in $\mathcal{L}_{RE}$ sind, dann ist $L \in \mathcal{L}_R$:

Beweis: Da per Annahme $L, L^C \in \mathcal{L}_{RE}$ sind, gibt es TMs M, M^C. Aus diesen beiden konstruieren wir nun eine TM A, welche L entscheidet und immer hält. Für eine Eingabe $w \in \Sigma^*$ simuliert A abwechselnd einen Schritt von M und M^C. Falls eine der beiden Simulationen endet, nimmt A den Output und gibt das Entsprechende aus (falls M^C hält einfach noch invertieren).

Falls $w \in L$, dann endet M nach endlicher Zeit und somit auch A. Falls $w \notin L$, dann ist $w \in L^C$ und somit endet M^C auch nach endlicher Zeit (und somit wieder auch A). Also endet A in jedem Fall in endlicher Zeit.

$\Rightarrow$ Für was braucht man das? Falls $L \notin \mathcal{L}_R$, dann muss $L^C \in \mathcal{L}_{RE}$ sein

2.3 Klassifizierung von verschiedenen Sprachen



2.4 Satz von Rice

Semantisch nichttriviales Entscheidungsproblem: Eine Sprache $L \subseteq$KodTM heisst s.n.E. falls:

1. Es gibt TM M_1 so dass Kod(M_1)$\in L$.
2. Es gibt TM M_2 so dass Kod(M_2)$\notin L$
3. Falls $L(A) = L(B)$ für TMs A, B, dann ist Kod(A)$\in L \iff$ Kod(B)$\in L$

$\Rightarrow$ Satz von Rice: Jedes semantisch nichttriviale Entscheidungsproblem ist unentscheidbar
$\Rightarrow$ Zum Beispiel ist die Sprache $\{Kod(M)|L(M) = L\}$ für alle Sprachen L unentscheidbar.

3 Übungen

3.1 S19, 17

Zeige, dass $|[0,1]| \geq \mathcal{P}(\{0,1,2\}^*)$:

Wir müssen gemäss Definition eine injektive Abbildung von $\mathcal{P}(\{0,1,2\}^*)$ nach $[0,1]$ angeben.

Sei $P \subseteq \{0,1,2\}$ eine beliebige Menge. Wir können P als einen unendlichen Bitvektor $b_1 b_2 \dots$ darstellen, wo $b_i = 1$ gdw. das kanonisch i-te Wort über Σ in P enthalten ist.

Wir können also für die Menge $\mathcal{B}$ aller unendichen Bitvektoren zeigen, dass gilt $\mathcal{B} \leq |[0,1]|$. Dazu (gleich wie im Buch) sei

$$f(b_1 b_2 \dots) := \sum_{i=1}^{\infty} b_i \cdot 10^{-i}$$

in anderen Worten: wir interpretieren die Folge $b_1 b_2 \dots$ als Dezimalbruch $0.b_1 b_2 \cdots \in [0,1]$. Die Injektivität ist dabei offensichtlich, da wir keine "Kollissionen" haben (anders als wenn wir bspw. auch Zahlen wie $0.\bar{9}$ betrachten würden!)

3.2 S19, 19a: $L_{\text{diag}} \leq_R$ Ł$_H$

Sei A ein Algorithmus, der L_H entscheidet. Wir wollen einen Algorithmus A_{diag} "bauen", welcher A als Subroutine verwendet und L_{diag} entscheidet. A_{diag} erhält die Eingabe $w \in \{0,1\}^*$ und gibt "1" aus, falls $w \in L_{\text{diag}}$.

Für eine Eingabe $w \in \{0,1\}^*$ berechnet A_{diag} zuerst $i \in \mathbb{N}$ so dass $w = w_i$. Dann berechnet A_{diag} die Kodierung der i-ten TM M_i. Daruas bastelt A_{diag} dann den String $M_i \# w = M_i \# w_i$ und übergibt das der "Subroutine" A als Eingabe.

Nach endlicher Zeit kriegen wir eine Anwort von A, nämlich "1" gdw M_i auf w_i haltet. Falls A zurückgibt, dass M_i haltet, wissen wir, dass wir die Arbeit von M_i in *endlicher* Zeit auf w simulieren können und wir finden raus, ob $w \in L(M_i)$ ist oder nicht. Das entsprechende Resultat geben wir dann aus (i.e. "$w \in L$" falls $w_i \in L(M_i)$).

Falls A hingegen sagt, dass M_i unendlich auf w_i rechnet, so ist $w_i \notin L(M_i)$ per Definition und wir geben $w \notin L_{\text{diag}}$ aus.

3.3 S19, 20a: $L_U \leq_{\mathbf{EE}} L_{\mathbf{HH}}$

Sei $L_{HH} = \{\text{Kod}(M_1)\#\text{Kod}(M_2)\#w \mid M_1, M_2 \text{ halten auf } w\}$. Zeige $L_U \leq_{EE} L_{HH}$:
Gesucht ist also ein Algorithmus F, welcher eine Eingabe x für L_U in eine Eingabe $f(x)$ für L_{HH} umwandelt:

1. **Immer** zuerst die Form prüfen! F entscheidet also, ob eine Eingabe x von der Form $\text{Kod}(M)\#w$ ist. Falls nicht, gibt F λ aus (da $\lambda \notin L_{HH}$).
2. Falls $x = \text{Kod}(M)\#w$, dann berechnet F' eine modifizierte Version M' von M: Alle Transitionen, welche von M aus in den verwerfenden Zustand führen, werden in M' in eine Endlosschleife umgeleitet.
3. Dann gibt F die Ausgabe $\text{Kod}(M')\#\text{Kod}(M')\#w$ aus.
4. Wir müssen nun noch zeigen, dass diese Reduktion korrekt ist, also $x \in L_U \iff F(x) \in L_{HH}$ für alle Eingaben:
5. Sei zuerst $x \in L_U$, dann ist $x = \text{Kod}(M)\#w$ und M akzeptiert w. Da F nur die Transition zu q_{reject} verändert, akzeptiert auch die modifizierte TM M' die Eingabe w. Insbesondere hält M' dann logischerweise auf w und somit ist $\text{Kod}(M')\#\text{Kod}(M')\#w \in L_{HH}$
6. Wir könnten nun zeigen, dass für eine Eingabe $F(x) \in L_{HH}$ gilt, dass $x \in L_U$ sein muss. Oftmals ist es aber viel leichter, das indirekt zu zeigen:

   Sei $x \notin L_U$. Dann unterscheiden wir zwei Fälle:

   - x ist nicht von der korrekten Form: Dann gibt F λ aus, welches offenbar nicht in L_{HH} ist.
   - Andernfalls ist $x = \text{Kod}(M)\#w$ und $w \notin M$. Dann endet die Berechnung auf w entweder im Zusatnd q_{reject} oder gar nicht. In beiden Fällen endet die Berechnung in M' daher nicht (per Konstruktion) und somit ist $\text{Kod}(M')\#\text{Kod}(M')\#w \notin L_{HH}$.

3.4 S19 / 16b)

Wir betrachten das Hilbert-Hotel. Angenommen, dieses ist leer und wir wollen keinen Gast in ein anderes Zimmer umziehen lassen. Zu einer beliebigen Zeitpunkt $t \in \mathbb{N}$ kommt eine Gruppe mit einer abzählbaren Anzahl an Gästen (welche nummeriert sind). Wie kann man die Gäst dann auf die Zimmer verteilen?

Wir wissen, dass es unendlich viele Primzahlen gibt. Ausserdem wissen wir, dass eine Zahl eindeutig durch ihre Primfaktorzerlegung bestimmt ist.

Sei daher p_i die i-te Primzahl. Wir definieren dann $g_i = \{p_i^j \mid j \in \mathbb{N}\}$ als die i-te Gruppe für alle i.
Wenn eine neue Gruppe an Tag t ankommt, teilen wir dann einfach die Gäste auf eine noch freie Gruppe g_k auf.
Korrektheit folgt aus Primfaktorzerlegung und Unendlichkeit der Primzahlen.

4 Neue Serie

- Bei Diagonalisierung:Das wichtigste ist, dass wir ein Wort finden, welches von keiner einzigen TM erkannt werden kann $\Rightarrow$ in der Kontraposition müssen wir also über *alle* TMs argumentieren.
- Aufgabe 19 habt ihr vermutlich schon so oder ähnlich in DiskMath gesehn $\Rightarrow$ Wie hängt das mit Aufzählbarkeit zusammen?