GZ Theoretische Informatik (WS16/17)

Lösungsvorschlag zu Aufgabenblatt 10

Aufgabe 1

Wir sollen eine Turingmaschine M skizzieren, die ihre eigene Kodierung rückwärts geschrieben ausgibt. Die einfachste Lösung ist, die Konstruktion aus dem Beweis des Rekursionssatz zu verwenden. Dabei müssen wir die Funktion t bzw. die Turingmaschine T aus dem Rekursionsatz entsprechend wählen. Sei also T eine Turingmaschine, die ihr erstes Argument rückwärts geschrieben ausgibt und ihr zweites Argument ignoriert:

$$u\$v \xrightarrow{T} u^R$$

Sei $Preprint_z$ eine Turingmaschine, die den String z vor ihre Eingabe schreibt:

$$x \xrightarrow{Preprint_z} zx$$

Man kann sich leicht überlegen, dass es auch eine Turingmaschine gibt, die für eine Eingabe z die Kodierung von $Preprint_z$ berechnet. Sei also die Turingmaschine B wie folgt:

$$z\$w \xrightarrow{B} \langle Preprint_{z\$}\rangle \# z\$w$$

Seien nun die Kodierungen von T und B durch $\langle T\rangle$ und $\langle B\rangle$ gegeben und sei weiter A eine Turingmaschine, die $Preprint_{\langle B\rangle\#\langle T\rangle\$}$ entspricht:

$$x \xrightarrow{A} \langle B\rangle\#\langle T\rangle\$x$$

Dann können wir die gesuchte Turingmaschine M kodieren als $\langle A\rangle\#\langle B\rangle\#\langle T\rangle$, das heißt M hat folgendes Verhalten:

$$x \xrightarrow{A} \langle B\rangle\#\langle T\rangle\$x \xrightarrow{B} \underbrace{\underbrace{\langle Preprint_{\langle B\rangle\#\langle T\rangle\$}\rangle}_{\langle A\rangle} \#\langle B\rangle\#\langle T\rangle\,\$x}_{\langle M\rangle} \xrightarrow{T} \langle M\rangle^R$$

Also insgesamt:

$$x \xrightarrow{M} \langle M\rangle^R$$

Aufgabe 2

Sei

$$\text{MIN1} := \{u \in \{0,1\}^* |\ \forall w \in \{0,1\}^* : L(M_u) = L(M_w) \Rightarrow \#_1(u) \leq \#_1(w)\}.$$

Zeige das MIN1 nicht entscheidbar ist.

Stelle zunächst fest: Es gibt unendlich viele Turing-entscheidbare Sprachen. Für jede dieser Sprachen gibt es eine Turing-Maschiene deren Beschreibung minimal viel 1en enthält, also ist MIN1 unendlich.

Angenommen MIN1 ist rekursiv aufzählbar. Definiere eine Turing-Maschiene M: Zähle MIN1 auf. Wegen dem Rekursionssatz kennen wir Beschreibung $\langle M\rangle$ von M. Da MIN1 unendlich ist und jede Transition in unserer Codierung 1en enthält, gelangt die Aufzählung nach endlich vielen Schritten zu einer Turing-Maschiene $M' \in$ MIN1 mit

$$\#_1(\langle M\rangle) < \#_1(\langle M'\rangle)$$

Simuliere M' auf M. Dann gilt

$$L(M) = L(M').$$

Aber daraus folgt $M' \notin$ MIN1, Widerspruch!

Aufgabe 3

1. L r.e. $\Longrightarrow \overline{L}$ r.e.
   Diese Aussage stimmt nicht.

   Gegenbeispiel: SAM ist r.e. und $\overline{SAM}$ ist nicht r.e.. Beides wurde in der Vorlesung bewiesen und deshalb kann die Implikation nicht stimmen.

2. L entscheidbar $\Longrightarrow \overline{L}$ entscheidbar
   Diese Aussage stimmt.

   Bei einer Turing Maschine die L entscheidet müssen nur die Zustände für JA bzw. NEIN vertauscht werden um $\overline{L}$ zu entscheiden

   Alternative: L entscheidbar $\Longleftrightarrow$ L akzeptierbar und $\overline{L}$ akzeptierbar $\Longleftrightarrow$ $\overline{L}$ akzeptierbar und $\overline{\overline{L}} = L$ akzeptierbar $\Longleftrightarrow$ $\overline{L}$ akzeptierbar.

3. L entscheidbar und $\overline{L}$ nicht r.e. $\Longrightarrow$ L regulär
   Diese Aussage ist richtig.

   L entscheidbar $\Longleftrightarrow$ L akzeptierbar und $\overline{L}$ akzeptierbar. Das bedeutet, daß die linke Seite der Implikation (L entscheidbar und $\overline{L}$ nicht r.e.) stets falsch ist. Damit ist die Implikation richtig. („Aus etwas falschem folgt beliebiges.“)

4. $L \leq L'$ und L r.e. $\Longrightarrow$ L' r.e.
   Diese Aussage stimmt nicht.

   Intuition: L ist „leichter“ als L' und deshalb kann man nicht schliessen, dass L' r.e. ist, auch wenn L auf L' reduzierbar ist.

   Wir widerlegen die Behauptung durch Angabe eines Gegenbeispiels. Wir können SAM auf die Sprache $L = \{u\$v | u \in SAM \text{ und } v \in \overline{SAM}\}$ reduzieren mit der Funktion $f(x) = x\$ < M >$ mit M irgendeine Turingmaschine für die gilt $L(M) \in \overline{SAM}$. Offensichtlich gilt $x \in SAM \iff x\$ < M > \in L$. Es bleibt zu zeigen, dass L' nicht r.e. ist: Dazu reduzieren wir $\overline{SAM}$ auf L mit $f(x) =< M_1 > \$x$ mit M_1 eine beliebige Turingmaschine für die gilt $L(M_1) \in SAM$. Die Argumentation ist analog zu oben.

5. $L \leq L'$ und L' r.e. $\Longrightarrow$ L r.e.
   Die Aussage ist richtig.

   L' sei akzeptierbar, d. h. es gibt eine Turing-Maschine M_2, die L' akzeptiert. $L \leq L'$ bedeutet, dass es eine berechenbare Funktion f gibt mit $x \in L \Longleftrightarrow f(x) \in L'$. Diese werde realisiert durch die Turing-Maschine M_1. Wir konstruieren daraus eine Turing-Maschine M, die L akzeptiert. Es sei eine Eingabe x gegeben. Zuerst simuliert M die Maschine M_1 und berechnet $f(x)$ (M_1 terminiert für alle $x \in L$), danach simuliert sie M_2 bei Eingabe $f(x)$. Sie akzeptiert $f(x)$ genau dann, wenn $f(x) \in L'$ ist, was gleichbedeutend mit $x \in L$ ist. Also akzeptiert M eine Eingabe x genau dann, wenn $x \in L$.

6. $L \leq L'$ und L nicht r.e. $\Longrightarrow$ L' nicht r.e.
   Diese Aussage ist richtig.

   Nehmen wir an, L' wäre akzeptierbar. Dann folgt aus Implikation 5, dass L akzeptierbar wäre, was aber der Voraussetzung widerspricht.

7. L regulär $\Longrightarrow$ L entscheidbar
   Die Aussage ist richtig.

   Für eine reguläre Sprache L existiert ein endlicher Automat, der L akzeptiert. Dieser kann von einer Turing-Maschine simuliert werden. Also ist L akzeptierbar. Bei den Abschlußeigenschaften für reguläre Sprachen hatten wir gezeigt, daß mit L auch $\overline{L}$ regulär ist. Also ist auch $\overline{L}$ akzeptierbar. Damit ist L entscheidbar.

8. L kontextfrei $\Rightarrow$ L r.e.
   Diese Aussage ist wahr.

   Beweis: Falls L kontextfrei ist, existiert ein Kellerautomat, der L akzeptiert. Somit gibt es auch eine Turing-Maschine, die dies tut und insbesondere folgt die Behauptung.

9. L kontextfrei $\Rightarrow$ L entscheidbar
   Diese Aussage ist wahr.

   Beweis: Im vorherigen Teil (h) wurde gezeigt: L kontextfrei $\Rightarrow$ L r.e., außerdem wurde bereits in der Vorlesung gezeigt: L entscheidbar $\Leftrightarrow$ L r.e. $\wedge$ $\overline{L}$ r.e. Um beides kombinieren zu können bleibt z.Z.: L kontextfrei $\Rightarrow$ $\overline{L}$ r.e.
   Dazu zunächst eine intuitive Betrachtung: Sowohl reguläre als auch DKA-Sprachen sind unter Komplementbildung abgeschlossen, d.h. L DKA-Sprache $\Rightarrow$ $\overline{L}$ DKA-Sprache $\overline{L}$ r.e. Für NKA-Sprachen und somit für kontextfreie Sprachen allgemein ist dies nicht der Fall. Der Grund ist intuitiv folgender: Ein nichtdeterministischer Kellerautomat akzeptiert ein Wort genau dann, wenn es eine akzeptierende Übergangsfolge für

dieses Wort gibt. Dabei ist es irrelevant, ob es beliebig viele weitere, nicht akzeptierende Übergangsfolgen für dieses Wort gibt. Genau deshalb funktioniert der Trick, der bei DEAs und DKAs zur Komplementbildung benutzt wird aber nicht mehr: dort werden normale und Endzustände getauscht, was dazu führt, daß der betreffende Automat die komplementäre Sprache akzeptiert. Beim NKA bedeutet dies: Wenn es einen akzeptierenden und einen nicht akzeptierenden Übergang gibt, akzeptieren sowohl der normale Automat, als auch der Automat mit vertauschten Zuständen beide das Wort. Was man bräuchte wäre ein Automat, der ein Wort explizit ablehnt, wenn es auch nur einen nicht akzeptierenden Übergang gibt, statt einem Automaten, der es akzeptiert, wenn auch nur ein akzeptierender Übergang existiert. Bei Turing-Maschinen mit expliziten Ja- und Nein-Endzuständen ist aber genau dies möglich und damit wird auch diese Sprache akzeptierbar.
Formal lässt sich dies jedoch auch über Grammatiken zeigen: Für jede kontextfreie Grammatik existiert eine (kontextfreie) Grammatik in Chomsky Normalform (Beweis siehe z.B. Uwe Schöning - Theoretische Informatik kurzgefasst Seite 53). Bei einer Grammatik in Chomsky Normalform haben alle Regeln eine der folgenden beiden Formen:
$A \to BC$
$A \to a$
Durch Anwendung einer Regel der ersten Form erhöht sich die Anzahl der Variablen um 1, durch Anwendung einer Regel der zweiten Form ersetzt man eine Variable durch ein Terminalsymbol. Mit einer solchen Grammatik kann man ein Wort der Länge n also in $2n-1$ Schritten ableiten – Ausgehend von einem Startsymbol kann man durch $n-1$ Anwendungen von Regeln der ersten Form n Variablen erzeugen, die man dann durch n Anwendungen von Regeln der zweiten Form in Terminale umwandelt. Nun kann man alle Wörter der Länge n bestimmen, die von der Grammatik erzeugt werden können und mit dem Eingabewort vergleichen. Es gibt maximal $|\Sigma|^n$ viele Wörter, für die man jeweils genau $2n-1$ Ableitungsregeln anwenden muß. Beachtet man nun noch, daß eine Grammatik nur endlich viele Produktionsregeln haben kann, ist das für ein gegebenes Wort von beliebiger aber konstanter Länge n eine Konstante (wenn auch eine große). Daraus folgt die Behauptung.

10. $\mathrm{L} = \{\langle \mathrm{M} \rangle \mid \mathrm{L(M)}$ ist kontextfrei $\} \Rightarrow$ L ist entscheidbar
Diese Aussage ist falsch. Dies folgt sofort aus dem Satz von Rice.

Aufgabe 4

Definiere f, g als:

$$f(n) = n$$
$$g(n) = \begin{cases} 1 & n = 2k \\ n^2 & n = 2k+1 \end{cases}$$

f, g sind fast überall positiv, aber $f \notin \mathcal{O}(g)$ und $g \notin \mathcal{O}(f)$ ($\Rightarrow g \notin o(f) \Rightarrow f \notin \omega(g)$).

Beweise:

- Angenommen $f \in \mathcal{O}(g)$. Dann existieren n_0, c, sodass $\forall n \geq n_0 : f(n) \leq c \cdot g(n)$, d.h. insbesondere für gerades n:

$$f(n) \leq c \cdot g(n)$$
$$n \leq c \cdot 1$$
$$n \leq c$$

  Widerspruch dazu, dass $c \in \mathbb{N}$ eine Konstante ist.

- Sei umgekehrt $g \in \mathcal{O}(f)$. Dann existieren n_0, c, sodass $\forall n \geq n_0 : g(n) \leq c \cdot f(n)$, d.h. insbesondere für ungerades n:

$$g(n) \leq c \cdot f(n)$$
$$n^2 \leq c \cdot n$$
$$n \leq c$$

Widerspruch dazu, dass $c \in \mathbb{N}$ eine Konstante ist.

Bemerkung: man kann auch dann Gegenbeispiele finden, wenn die zusätzliche Einschränkung gilt, dass beide f, g monoton steigend sein müssen:

$$f(n) = \begin{cases} 1 & n = 0 \\ f(n-1) & n = 2k+1 \\ n \cdot g(n-1) & n = 2k, k > 0 \end{cases} \quad g(n) \quad = \begin{cases} 0 & n = 0 \\ n \cdot f(n-1) & n = 2k+1 \\ g(n-1) & n = 2k, k > 0 \end{cases}$$

f, g sind monoton steigend, aber $f \notin \mathcal{O}(g)$ und $g \notin \mathcal{O}(f) (\Rightarrow g \notin o(f) \Rightarrow f \notin \omega(g))$.

Beweise:

- Angenommen $f \in \mathcal{O}(g)$. Dann existieren n_0, c, sodass $\forall n \geq n_0 : f(n) \leq c \cdot g(n)$, d.h. insbesondere für gerades n:

$$\begin{aligned} f(n) &\leq c \cdot g(n) \\ n \cdot g(n-1) &\leq c \cdot g(n-1) \\ n &\leq c \end{aligned}$$

  Widerspruch dazu, dass $c \in \mathbb{N}$ eine Konstante ist.

- Sei umgekehrt $g \in \mathcal{O}(f)$. Dann existieren n_0, c, sodass $\forall n \geq n_0 : g(n) \leq c \cdot f(n)$, d.h. insbesondere für ungerades n:

$$\begin{aligned} g(n) &\leq c \cdot f(n) \\ n \cdot f(n-1) &\leq c \cdot f(n-1) \\ n &\leq c \end{aligned}$$

  Widerspruch dazu, dass $c \in \mathbb{N}$ eine Konstante ist.