Theoretische Informatik (WS05)
Prof. Dr. Raimund Seidel
Wei Ding
Lösungsvorschlag zu Aufgabenblatt 8

Aufgabe 1

Behauptung 1 A ist *lexikographisch rekursiv aufzählbar* $\Rightarrow$ A ist *entscheidbar*.

Beweis: nach der Definition heißt eine Sprache A *lexikographisch rekursiv aufzählbar* entweder wenn sie *endlich* ist oder wenn es eine totale, berechenbare Funktion $G : \mathbb{N} \to \Sigma^*$ gibt, die A lexikographisch aufzählt.

Im ersten Fall ist die Sprache A klar entscheidbar, da sie endlich und dann regulär ist. Wir brauchen die Macht einer DTM nicht; ein DEA reicht aus, um die Sprache A zu entscheiden.

Im zweiten Fall existiert eine DTM M, die die Funktion G berechnet. Mithilfe dieser DTM M konstruieren wir eine DTM N, die die Sprache A entscheidet: auf Eingabe x initialisiert die DTM N einen Zähler n mit 0, d.h. schreibt die Zahl 0 in Binärdarstellung auf ein Arbeitsband und simuliert die DTM M auf dieser Eingabe. Dann vergleicht N *lexikographisch* die Ausgabe $y = G(n)$ der DTM M mit der Eingabe x. (Das Problem, ob ein Wort x lexikographisch kleiner, gleich, oder größer ist als ein Wort y, ist natürlich entscheidbar.)

- Ist $x = y$, dann hält N und akzeptiert die Eingabe.
- Ist $x > y$, dann hält N und verwirft die Eingabe.
- Ist $x < y$, dann erhöht N den Zähler n um eins und wiederholt den Prozess, d.h. sie lässt wieder M auf n laufen usw.

```
for (n=0; G(n)<x; ++n)
;
return (G(n)==x);
```

Der Prozess ist offensichtlich korrekt und *terminiert* immer, da $x \in A \Rightarrow \exists n \in \mathbb{N} : G(n) = x$ und dieses n wird in der $(n+1)$-ten Wiederholung der Schleife gefunden und damit hält die DTM N an und akzeptiert.

Auf der anderen Seite, $x \notin A \Rightarrow \exists n_0 \in \mathbb{N} : \forall n \in \mathbb{N}, n < n_0 : G(n) < x \wedge \forall n \in \mathbb{N}, n \geq n_0 : G(n) > x$. Dieses n_0 wird wieder in der $(n_0 + 1)$-ten Wiederholung des Prozesses gefunden und damit wird die DTM N anhalten und abhlehnen.

Da die DTM N immer terminiert und genau dann die Eingabe x akzeptiert, wenn $x \in A$, ist die Sprache A entscheidbar.

Behauptung 2 A ist *entscheidbar* $\Rightarrow$ A ist *lexikographisch rekursiv aufzählbar*.

Beweis: nach Definition heißt eine Sprache A *entscheidbar*, wenn es eine DTM M gibt, die immer terminiert und genau dann die Eingabe x akzeptiert, wenn $x \in A$.

Wenn die Sprache A endlich ist, dann gibt es nichts zu zeigen, da A *nach Definition* lexikographisch rekursiv aufzählbar ist.

Wenn jetzt A unendlich ist, dann konstruieren wir mithilfe der DTM M eine DTM N, die A lexikographisch rekursiv aufzählt: auf Eingabe $bin(n)$, wobei $n \in \mathbb{N}$, initialisiert die DTM N einen Zähler m mit 0, d.h. notiert die Zahl 0 in Binärdarstellung auf ein Arbeitsband und schreibt das erste Wort der lexikographischen Ordnung von Σ^*, d.h. das leere Wort ϵ, auf ein anderes Arbeitsband. Nennen wir den Inhalt dieses letzten Arbeitsbands w. Folglich wird die DTM M auf Eingabe w simuliert. Da die DTM M die Sprache A *entscheidet* wird sie nach *endlich vielen* Schritten terminieren und die Eingabe w genau dann akzeptieren, wenn $w \in A$ gilt. Wenn M akzeptiert, dann wird der Zähler m mit der Eingabe n verglichen. Wenn $m = n$, dann gilt $w = G(n)$ und N kann jetzt w ausgeben und terminieren. Wenn immer noch $m < n$, dann wird der Zähler m um eins erhöht, aus w das *lexikographisch* nächste Wort von Σ^* berechnet (die Funktion, die den lexikographischen Nachfolger ihres Eingabeworts ausgibt, ist natürlich berechenbar) und der Prozess wiederholt, d.h. die DTM M wird auf w simuliert usw.

```
for (m=0, w=$\epsilon$; ; w=lexikographischer_Nachfolger_von(w))
  if (M(w))
    if (m==n)
      return w;
    else
      ++m;
```

Der Prozess ist korrekt, da jedes Mal w von M akzeptiert wird, $w = G(m)$ gilt und w wird genau dann ausgegeben, wenn $m = n$ gilt, d.h. wenn $w = G(n)$.

Der Prozess terminiert immer, da die Sprache A unendlich ist und, folglich $\forall n \in \mathbb{N} : G(n)$ existiert und nach endlich vielen Schritten gefunden wird.

Aufgabe 2

Betrachten wir zwei total Turing berechenbare Funktionen $f : \Sigma^* \to \Gamma^*$ und $g : \Gamma^* \to \Pi^*$, so können wir beobachten, dass offensichtlich auch die Hintereinanderausführung $g \circ f : \Sigma^* \to \Pi^*$ total Turing berechenbar ist: Werden f und g durch die Turingmaschinen M und N berechnet, dann wird $g \circ f$ von der Maschine berechnet, die bei jeder Eingabe zunächst M simuliert, bis diese in einen Haltezustand gelangt, und auf dem so erzeugten Zwischenergebnis wie N fortfährt.

Seien nun die Sprachen $A \subset \Sigma^*$, $B \subset \Gamma^*$ und $C \subset \Pi^*$ gegeben mit $A \preceq B$ und $B \preceq C$. Dann existieren total Turingberechenbare Funktionen $f : \Sigma^* \to \Gamma^*$ und $g : \Gamma^* \to \Pi^*$ mit der Eigenschaft, dass $x \in A \Leftrightarrow f(x) \in B$ und $y \in B \Leftrightarrow g(y) \in C$.

Nach obiger Beobachtung ist die Funktion $h = g \circ f$ total Turing berechenbar und es gilt:

$$x \in A \Leftrightarrow f(x) \in B \Leftrightarrow g(f(x)) = h(x) \in C$$

Damit ist also A reduzierbar auf C : $A \preceq C$, und die Relation ist transitiv

Theoretische Informatik (WS05)
Prof. Dr. Raimund Seidel
Wei Ding
Lösungsvorschlag zu Aufgabenblatt 8

Aufgabe 3

1. $MAQ = \{\mathcal{M}\$w\$q : M$ erreicht auf Eingabe w vom Startzustand irgendwann den Zustand $q\}$
   Betrachten wir nun folgende Instanz von *MAQ* :
   $MAQ_h = \{\mathcal{M}\$w\$h : M$ erreicht auf Eingabe w vom Startzustand irgendwann den Haltezustand$\}$.
   Annahme: MAQ_h ist Turing–entscheidbar, d.h. es existiert eine TM M_{MAQ_h}, die MAQ_h entscheidet. Wir ändern M_{MAQ_h} nun so ab, daß mit der Modifikation das Halteproblem L_{UNIV} entschieden werden könnte:
   Sei M_{wr} die TM, die bei Eingabe $\mathcal{M}\$w$, die Kodierung des Haltezustands h vom M auf das Band schreibt, also $\mathcal{M}\$w\$h$ als Ausgabe liefert, dann liefert die Kombination von M_{wr} und M_{MAQ_h} einen Algorithmus, der L_{UNIV} entscheidet, was zu einem Widerspruch führt. Folglich ist MAQ_h und damit das allgemeinere Problem MAQ nicht Turing–entscheidbar !

2. Es ist Turing–entscheidbar, ob eine beliebige (oBdA deterministische) TM M mit Eingabe w je ihren Kopf nach links rückt.
   Beweis : Wir simulieren M auf Eingabe w. Sei (ε, s, w) die Startkonfiguration. Wenn M nach links geht, dann antworten wir mit ja, falls nicht, so wird M ein Zeichen $a \in \Sigma$ auf das Band schreiben und auf dem Feld bleiben oder um ein Feld nach rechts rücken. Wir merken uns nun den gemachten Rechenschritt. Aufgrund des endlichen Alphabets und der endlichen Zustandsmenge kann M nur endlich viele verschiedene Rechenschritte (ohne Linksbewegung) machen, dann muß sich ein Rechenschritt wiederholen und wir antworten mit nein, da nun eine Schleife erreicht wurde und folglich keine Linksbewegung mehr stattfinden kann (wegen des Nichtdeterminismus).

3. - **M**it Satz von Rice
     Nehmen wir an, wir könnten entscheiden, ob für zwei beliebige TM M_1 und M_2 die Gleichheit $L(M_1) = \overline{L(M_2)}$ gilt (dieses Problem sei als MAK bezeichnet), dann könnte man beispielsweise M_1 als die triviale Maschine, die jeden Input verwirft (d.h. $L(M_1) = \emptyset$), fixieren und hätten somit einen Algorithmus, welcher entscheidet, ob die TM M_2 Σ^* akzeptiert. Dies ist bekanntlicherweise nicht entscheidbar (Problem MAA). Folglich ist MAK nicht Turing–entscheidbar !
   - **O**hne Satz von Rice
     Nehmen wir an, das Problem sei entscheidbar. Dann können wir das Halteproblem wie folgt entscheiden: auf Eingabe $enc(M), x$, wobei $enc(M)$ eine anständige Kodierung einer DTM M ist und x eine Eingabe zur M, konstruieren wir eine DTM M_x, die zunächst überprüft ob ihre Eingabe gleich x ist. Wenn ihre Eingabe ungleich x ist, dann divergiert die DTM M_x. Wenn ihre Eingabe gleich x ist, dann simuliert M_x die DTM M, und falls M hält, akzeptiert (egal ob M akzeptiert).

     ```
     if (input == x) then {M(input); accept} else diverge
     ```

     D.h., wenn M auf X hält, dann gilt $L(M_x) = \{x\}$ und wenn M auf x nicht hält, dann gilt $L(M_x) = \{\}$. Die Konstruktion von M_x aus M ist natürlich algorithmisch machbar.
     Des weiteren konstruieren wir eine DTM N_x, die alle Wörter außer x akzeptiert.

     ```
     if (input == x) then diverge else accept
     ```

D.h., $L(N_x) = \Sigma^* - \{x\}$. Die Konstruktion von N_x aus x ist algorithmisch sehr einfach. Jetzt gilt, dass M auf Eingabe x genau dann hält, wenn $L(M_x) = \overline{L(N_x)}$ gilt.