Übungsblatt03 - Spezielle unentscheidbare Sprachen - Musterlösung

TH Mittelhessen, FB MNI, Berechenbarkeit und Komplexität, Prof. Dr. B. Just

Aufgabe 1

Bitte beweisen Sie mit einem Diagonalisierungsverfahren, dass das Halteproblem unentscheidbar ist:

$$\text{HALT}_{\text{TM}} = \{(M, \omega) : \text{Turingmaschine } M \text{ hält bei Input } \omega\}.$$

Die Methode der Diagonalisierungsverfahren ist nach dem Cantor'schen Diagonalisierungsverfahren (1877) benannt. Diagonalisierungsverfahren beweisen Nichtexistenzen durch Widerspruch. Sie nehmen die Existenz an, und führen sie zu einer paradoxen Situation vom folgenden Typ herbei:
„Wenn das Objekt eine bestimmte Eigenschaft hat, hat es sie nicht, und umgekehrt."

Lösung Aufgabe 1:

Angenommen, es gibt eine Turingmaschine TM_{HALT}, die HALT_{TM} entscheidet.

TM_{HALT} erwartet also als Input die Beschreibung einer Turingmaschine M und einen Input ω für M. Hat die Bandinschrift nicht diese Form, verwirft TM_{HALT}. Ansonsten ist

$$TM_{\text{HALT}}(M, \omega) = \begin{cases} \text{accept}, & \text{falls } M, \text{ angewandt auf } \omega, \text{ anhält} \\ \text{reject} & \text{falls } M, \text{ angewandt auf } \omega, \text{ nicht anhält.} \end{cases}$$

TM_{HALT} hält also insbesondere immer an.

Wenn es TM_{HALT} gibt, gibt es auch eine Turingmaschine H_1, die als Input eine Turingmaschine M erwartet und dann entscheidet, ob diese, angewandt auf sich selbst, anhält.
Denn: H_1 prüft zunächst, ob der Input die Codierung einer Turingmaschine ist. Falls nicht, verwirft H_1.
Ansonsten kopiert H_1 den Input M noch einmal auf das Band, und ruft dann TM_{HALT} auf. TM_{HALT} entscheidet nun, ob M, angewandt auf sich selbst, anhält.
Es ist also

$$H_1(M) = \begin{cases} \text{accept}, & \text{falls } M, \text{ angewandt auf } M, \text{ anhält} \\ \text{reject} & \text{falls } M, \text{ angewandt auf } M, \text{ nicht anhält.} \end{cases}$$

Wenn es H_1 gibt, gibt es auch die Turingmaschine $\overline{H_1}$, die bei Input einer Turingmaschine M erfüllt:

$$\overline{H_1}(M) = \begin{cases} \text{accept}, & \text{falls } M, \text{ angewandt auf } M, \text{ nicht anhält} \\ \text{loop} & \text{sonst.} \end{cases}$$

Denn $\overline{H_1}$ kann programmiert werden, indem zunächst geprüft wird, ob der Input die Codierung einer Turingmaschine ist. Falls nicht, wird verworfen. Andernfalls wird H_1 aufgerufen. Falls H_1 verwirft, akzeptiert $\overline{H_1}$. Andernfalls verwirft H_1, und in diesem Fall geht $\overline{H_1}$ in eine Endlosschleife. Beachte, dass H_1 immer anhält, deshalb ist die Konstruktion von $\overline{H_1}$ möglich.

Wenn $\overline{H_1}$ auf sich selbst angewendet wird, erhält man

$$\overline{H_1}(\overline{H_1}) = \begin{cases} \text{accept}, & \text{falls } \overline{H_1}, \text{ angewandt auf } \overline{H_1}, \text{ nicht anhält} \\ \text{loop} & \text{sonst.} \end{cases}$$

D.h., $\overline{H_1}$ angewandt auf sich selbst akzeptiert, wenn $\overline{H_1}$ angewandt auf sich selbst, nicht anhält. Das ist ein Widerspruch.

Also kann es TM_{HALT} nicht geben. Q.e.d.

Aufgabe 2

Bitte beweisen Sie mit durch Reduktion des Halteproblems (siehe Aufgabe 1) auf die Sprache A_{TM}, dass diese unentscheidbar ist:

$$\text{A}_{\text{TM}} = \{(M, \omega) : \text{Turingmaschine } M \text{ akzeptiert Input } \omega\}.$$

Die Methode der Reduktion führt allgemein ein zu untersuchendes Berechnungsproblem auf ein bereits untersuchtes zurück. Es wird gezeigt, dass mit einem Algorithmus für das zu untersuchende Problem auch ein Algorithmus für das bereits untersuchte Problem existiert. Das kann dann zu einem Widerspruch zu bekannten Eigenschaften des bereits untersuchten Problems führen.
In dieser Aufgabe ist speziell zu zeigen: Wenn A_{TM} entscheidbar wäre, wäre auch das Halteproblem entscheidbar.

Lösung Aufgabe 2:

Als Vorarbeit definieren wir für jede Turingmaschine M eine Turingmaschine $\overline{M}$, die genau dann anhält, wenn M anhält, und dann das Gegenteil des Outputs von M ausgibt.
Es ist also für jeden Input ω

$$\overline{M}(\omega) = \begin{cases} \text{accept}, & \text{falls } M(\omega) = \text{reject}, \\ \text{reject} & \text{falls } M(\omega) = \text{accept} \\ \text{loop} & \text{falls } M(\omega) = \text{loop}. \end{cases}$$

$\overline{M}$ kann aus der Beschreibung von M gewonnen werden, indem man dort den akzeptierenden und den verwerfenden Haltezustand mit zwei neuen Zustandsnamen umbenennt, und von dort aus in den verwerfenden bzw. akzeptierenden Haltezustand geht.

Für den Beweis, dass A_{TM} unentscheidbar ist, nehmen wir nun an, das Gegenteil sei der Fall, A_{TM} sei also entscheidbar. Das heißt, es gibt eine Turingmaschine X, die zunächst prüft, ob, ob der Input die Form (M, ω) hat, verwirft, falls das nicht der Fall ist, und ansonsten erfüllt:

$$X(M, \omega) = \begin{cases} \text{accept}, & \text{falls } M(\omega) = \text{accept}, \\ \text{reject} & \text{falls } M(\omega) = \text{reject oder} M(\omega) = \text{loop}. \end{cases}$$

Da X ein Entscheider ist, hält X auf jedem Input an.

Mit X gibt es auch eine Turingmaschine Y, die prüft, ob der Input die Form (M, ω) für eine Turingmaschine M hat, in diesem Fall verwirft, und andernfalls erfüllt

$$Y(M, \omega) = \begin{cases} \text{accept}, & \text{falls } X(M, \omega) = \text{accept oder } X(\overline{M}, \omega) = \text{accept}, \\ \text{reject} & \text{sonst.} \end{cases}$$

Y erhält man, indem man auf den Input zunächst X anwendet, akzeptiert, falls X akzeptiert, und ansonsten $\overline{M}$ codiert und X noch einmal anwendet. Da X immer anhält, hält auch Y immer an.

Es ist dann aber nach Definition von $\overline{M}$

$$Y(M,\omega) = \begin{cases} \text{accept}, & \text{falls } X(M,\omega) = \text{accept oder } X(M,\omega) = \text{reject}, \\ \text{reject} & \text{sonst.} \end{cases}$$

Also entscheidet Y das Halteproblem HALT_{TM}. Dies ist nach Aufgabe 1 unmöglich, also kann es Y nicht geben, und somit ist auch A_{TM} unentscheidbar. Q.e.d.

Aufgabe 3

a.) Bitte zeigen Sie, dass die folgende Sprache unentscheidbar ist:

$$\text{E}_{\text{TM}} = \{M : M \text{ ist Turingmaschine mit } L(M) = \varnothing\}.$$

b.) Bitte zeigen Sie, dass die folgende Sprache unentscheidbar ist:

$$EQ_{\text{TM}} = \{(M_1, M_2) : \ M_1 \text{ und } M_2 \text{ sind Turingmaschinen, die dieselben Sprachen akzeptieren}\}.$$

Lösung Aufgabe 3:

a.) (Siehe Sipser, Seite 217)

Wir zeigen: Wäre E_{TM} entscheidbar, so wäre auch A_{TM} entscheidbar, ein Widerspruch zu Aufgabe 2.

Vorarbeit: Zu jeder gegebenen Turingmaschine M und einem Input ω für M kann man eine Turingmaschine M_ω konstruieren, die sich auf einem beliebigen Input ρ wie folgt verhält

$$M_\omega(\rho) = \begin{cases} M(\omega) & \text{falls } \rho = \omega \\ \text{reject} & \text{sonst.} \end{cases}$$

M_ω prüft, ob der Input mit ω übereinstimmt, verwirft, falls das nicht der Fall ist, und lässt ansonsten M auf ω laufen. Das Ergebnis kann accept, reject oder loop sein.

Wir betrachten die Menge $L(M_\omega)$. Diese ist entweder leer (falls M den Input ω nicht akzeptiert), oder sie besteht nur aus dem Input ω (falls M den Input ω akzeptiert).

Somit ist $(M,\omega) \in A_{\text{TM}}$ genau dann, wenn $M_\omega \notin E_{\text{TM}}$.

Wäre also E_{TM} entscheidbar, wäre auch A_{TM} entscheidbar:
Um $(M,\omega) \in A_{\text{TM}}$ zu entscheiden, entscheidet man $M_\omega \notin E_{\text{TM}}$. Dies ist ein Widerspruch, da A_{TM} nicht entscheidbar ist. Also ist auch E_{TM} nicht entscheidbar. Q.e.d.

b.) (Siehe Sipser, Seite 220)

Sei $T_\emptyset$ die Turingmaschine, die jeden Input verwirft.

Für eine beliebige Turingmaschine M gilt offenbar:

$$M \in E_{\text{TM}} \text{ genau dann, wenn } (M, T_\emptyset) \in EQ_{\text{TM}}.$$

Da E_{TM} nicht entscheidbar ist, kann auch EQ_{TM} nicht entscheidbar sein. Q.e.d.

Aufgabe 4

(Aus Sipser, Problem 4.12. S. 211)

> Sei L eine entscheidbare Sprache, die aus lauter Entscheidern besteht - also aus Turingmaschinen, die immer anhalten.
> Bitte zeigen Sie, dass es eine entscheidbare Sprache S gibt, die von keinem der Entscheider aus L entschieden wird.

Lösung Aufgabe 4:

Hab ich noch nicht - im Sipser ist der Hint, dass man einen Enumerator für A betrachten soll.

Viel Spass und Erfolg!