Dr. Sebastian Bab WiSe 12/13

Theoretische Grundlagen der Informatik für TI

Termin: VL 20 vom 17.01.2013

Halteproblem, Reduzierbarkeit und der Satz von Rice

- Mit der Diagonalisierungssprache haben wir eine Sprache kennengelernt, welche nicht semi-entscheidbar ist. Es stellt sich die Frage, ob es auch Sprachen gibt, welche semi-entscheidbar, aber nicht entscheidbar sind.

- $L_U = \{ (M, w) \mid M \text{ akzeptiert } w \}$, die sogenannte Universalsprache wird von einer universellen Turingmaschine erkannt. Daher muss L_U semi-entscheidbar sein. Wäre L_U komplett entscheidbar, so wäre auch $\overline{L_U}$ entscheidbar.

  Wir konstruieren nun eine Turingmaschine M_D folgendermaßen. Bei Eingabe des Wortes w erzeugt M_D als erstes das Tupel (M, w), wobei M die Turingmaschine ist, die durch w kodiert wird. Dieses Tupel wird nun als Eingabe für die Turingmaschine genutzt, die $\overline{L_U}$ entscheidet. Diese Entscheidung wird dann als Antwort von M_D ausgegeben.

  Die Sprache die von M_D berechnet wird ergibt sich folgendermaßen: Die Hilfsmaschine die $\overline{L_U}$ entscheidet, akzeptiert alle Tupel (M, w) bei denen die Maschine M das Wort w nicht akzeptiert. M_D benutzt diese Maschine und gibt als Eingabe das Tupel (M, w), so dass w die Maschine M kodiert. In $L(M_D)$ sind also alle Worte w, die nicht von der Turingmaschine akzeptiert werden, die durch w selbst kodiert wird. Die Sprache $L(M_D)$ ist also genau die Diagonalisierungssprache, von welcher wir wissen, dass sie nicht semi-entscheidbar ist. Es kann also die Maschine M_D nicht existieren. Daher kann keine Maschine für $\overline{L_U}$ existieren und somit kann die Universalsprache nicht entscheidbar sein.

- Ein weiteres bekanntes Problem ist das *spezielle Halteproblem*:

  $$K = \{ w \mid M_w \text{ hält bei Eingabe } w \}$$

  Hier geht es also nicht darum, dass das Wort w akzeptiert wird, sondern nur um die Frage, ob M_w bei Eingabe w stoppt, also nicht in einer Endlosschleife läuft. Das Spezielle Halteproblem ist semi-entscheidbar, da ein Semi-Entscheidungsalgorithmus leicht konstruierbar ist:

  Bei Eingabe w wird M_w simuliert. Sollte M_w akzeptieren, oder verwerfen, so wird w akzeptiert, da M_w gestoppt ist.

Wäre nun das Spezielle Halteproblem entscheidbar, so könnte man daraus den folgenden Widerspruch erzeugen: Die Maschine M_K die das spezielle Halteproblem entscheidet, wird zur Konstruktion einer anderen Maschine M genutzt, die folgendermaßen arbeitet: M leitet ein Eingabewort direkt an M_K weiter. Akzeptiert M_K das Eingabewort, so wird die Eingabe wieder an M_K geleitet, so dass eine Endlosschleife entsteht. Verwirft M_K hingegen die Eingabe, so akzeptiert M. Wird nun der Code der Maschine M als Eingabe von M verwendet ergibt sich der Widerspruch:

Angenommen, w sei der Code von M und M hält bei Eingabe von w. Dann muss M_K die Eingabe w verworfen haben. Dies bedeutet, dass $w \notin L(M_K)$ ist, also dass M bei Eingabe w nicht hält. Da dies ein Widerspruch zur Annahme ist, kann M_K nicht existieren. Das spezielle Halteproblem ist also nicht entscheidbar.

- Es wurden nun mehrfach Ergebnisse auf andere Ergebnisse zurückgeführt, wie zum Beispiel die Unentscheidbarkeit der Universalsprache auf die Unentscheidbarkeit der Diagonalisierungssprache. Diese Beziehung wird nun unter dem Begriff *Reduktion* formalisiert.

Definition 1 (Reduktion)
Seien $L_1 \subseteq \Sigma^*$ und $L_2 \subseteq \Gamma^*$ zwei Sprachen. Dann heißt L_1 auf L_2 reduzierbar, *geschrieben $L_1 \leq L_2$, wenn es eine totale, berechenbare Funktion $f : \Sigma^* \to \Gamma^*$ gibt, so dass für alle $x \in \Sigma^*$ gilt*

$$x \in L_1 \Leftrightarrow f(x) \in L_2.$$

Die Funktion f kann man sich dabei als Übersetzung eines Problems in ein anderes vorstellen, wobei man nun eine Problemstellung aus L_1 löst, indem man sie nach L_2 übersetzt und dort löst. Dies ist besonders dann nützlich, wenn man für L_2 einen Lösungsalgorithmus hat, aber nicht für L_1.

Satz 2
Falls $L_1 \leq L_2$ und L_2 ist entscheidbar (semi-entscheidbar), so ist auch L_1 entscheidbar (semi-entscheidbar).

Beweis:
Aus der Voraussetzung folgt, dass es eine totale und berechenbare Funktion f gibt, mit $x \in L_1 \Leftrightarrow f(x) \in L_2$. Da weiterhin L_2 entscheidbar (semi-entscheidbar) ist, ist die (halbe) charakteristische Funktion χ_{L_2} (bzw. χ'_{L_2}) berechenbar. Dann ist auch die (halbe) charakteristische Funktion $\chi_{L_1} = \chi_{L_2} \circ f$ (bzw. $\chi'_{L_2} = \chi'_{L_1} \circ f$) berechenbar und damit L_1 entscheidbar (semi-entscheidbar). □

Aus der Kontraposition zu diesem Satz ergibt sich, dass aus L_1 unentscheidbar und $L_1 \leq L_2$ auch L_2 unentscheidbar folgt.

- Das Halteproblem kann wie folgt verallgemeinert werden:

$$H = \{ (M, w) \mid M \text{ hält bei Eingabe } w \}.$$

Auch das (allgemeine) Halteproblem ist semi-entscheidbar, aber nicht entscheidbar.
Beweis:
Man kann das spezielle Halteproblem auf das allgemeine Halteproblem reduzieren. Wir zeigen also $K \leq H$, indem wir eine berechenbare Übersetzungsfunktion f angeben. Die Funktion f übersetzt dabei jedes Eingabewort w in das Tupel (M, w), so dass w der Code der Maschine M ist. Nun gilt

$$w \in K \Leftrightarrow f(w) = (M, w) \in H.$$

Wäre also das (allgemeine) Halteproblem entscheidbar, so wäre auch das spezielle Halteproblem entscheidbar. □

- Ein weiterer Spezialfall ist das Halteproblem auf leerem Band:

$$H_0 = \{ w \mid M_w \text{ hält bei Eingabe } \lambda \}.$$

Auch H_0 ist semi-entscheidbar aber nicht entscheidbar.

Beweis:
Wir zeigen $H \leq H_0$. Wir definieren eine totale berechenbare Funktion f die folgendermaßen arbeitet: Für jede Eingabe der Form (M, w) liefert f eine Turingmaschine M', die bei leerer Eingabe zuerst das Eingabewort w erzeugt und danach wie M arbeitet. Ist nun $(M, w) \in H$, so ist stoppt M bei Eingabe w. Daher stoppt auch M' bei leerer Eingabe und damit ist $f((M, w)) = M' \in H_0$. □

- Das Komplement des Halteproblems ist nicht semi-entscheidbar. Wäre es semi-entscheidbar, würde daraus folgen, dass das Halteproblem entscheidbar wäre.

Satz 3
(Satz von Rice) Sei $\mathcal{R}$ die Klasse aller Turing-berechenbaren Funktionen. Sei weiterhin $\mathcal{S}$ mit

$$\emptyset \subset \mathcal{S} \subset \mathcal{R}$$

eine beliebige Teilmenge von $\mathcal{R}$. Dann ist die Sprache

$$C(\mathcal{S}) = \{ w \mid \textit{die von } M_w \textit{ berechnete Funktion liegt in } \mathcal{S} \}$$

nicht entscheidbar.

Dies bedeutet, dass man algorithmisch nicht entscheiden kann, welche Art von Funktion durch eine Turingmaschine berechnet wird. Zum Beispiel ist die Menge aller Turingmaschinen, welche eine konstante Funktion berechnen unentscheidbar.

Beweis:
Sei $\Omega \in \mathcal{R}$ die vollständig undefinierte Funktion. Es gilt nun entweder $\Omega \in \mathcal{S}$ oder $\Omega \notin \mathcal{S}$.

Fall 1: $\Omega \in \mathcal{S}$
Wir zeigen nun, dass $\overline{H_0} \leq C(\mathcal{S})$ gilt und damit auch $C(\mathcal{S})$ nicht entscheidbar sein kann.

Da $\mathcal{R}$ berechenbar ist und $\mathcal{S} \subset \mathcal{R}$ ist, gibt es eine Funktion $q \in \mathcal{R} \setminus \mathcal{S}$ und eine Turingmaschine M_q die q berechnet. Wir beschreiben nun eine totale, berechenbare Übersetzungsfunktion f, die jedem Eingabewort $w \in \{0,1\}^*$ eine Turingmaschine M zuordnet, die folgendermaßen arbeitet: Bei Eingabe w wird zuerst M_w bei Eingabe λ simuliert. Hält M_w, so wird danach M_q bei Eingabe y simuliert.

Diese Maschine berechnet die folgende Funktion

$$g = \begin{cases} \Omega & M_w \text{ stoppt nicht bei Eingabe } \lambda \\ q & \text{sonst} \end{cases}$$

Gilt nun $w \in H_0$, dann hält M_w bei Eingabe λ und damit ist $g = q$. Die Maschine die durch $f(w)$ berechnet wird, berechnet also eine Funktion die nicht in $\mathcal{S}$ liegt und damit ist $f(w) \notin C(\mathcal{S})$.

Gilt allerdings $w \notin H_0$, so hält M_w bei Eingabe λ nicht an und daher gilt $g = \Omega$. Da $\Omega \in \mathcal{S}$ gilt auch $f(w) \in C(\mathcal{S})$. Es gilt also $\overline{H_0} \leq C(\mathcal{S})$.

Fall 2: $\Omega \notin \mathcal{S}$
Analog zu Fall 1 wird gezeigt, dass $H_0 \leq C(\mathcal{S})$ □