Theoretische Informatik 2
Ausarbeitung der Vorlesung vom Fr, 26.01.2001

Henryk Feider
(Matr.: 702899)

7 Aufzählbarkeit und Entscheidbarkeit

Bisher war uns nur die Entscheidbarkeit bekannt, wobei wir wußten, daß zum Beispiel das *Selbstanwendungsproblem* und das *Halteproblem* **nicht** entscheidbar sind.

Jetzt behandeln wir die **Aufzählung**, welche uns eine einseitige Entscheidbarkeit bei solchen Problemen ermöglicht.

Speziell der **Satz B**, dessen Beweis das zentrale Thema dieser Vorlesung war, sagt aus, daß bei einer gegebenen Menge *M* Aufzählbarkeit von *M*, *M* Ergebnisbereich einer Turing-Maschine, *M* Haltebereich einer Turing-Maschine, *M* partiell-rekusiv und *M*∈*Chomsky-0* äquivalent untereinander sind.

Definition A:

Sei *X* ein endliches Alphabet. Eine beliebige Menge $M \subseteq X^\star$ heißt *aufzählbar* wenn $M = \emptyset$ oder wenn es eine Turing-Maschine τ mit $\tau = (S, X, \Gamma, \delta, s_0, b)$ gibt.

1. $X \subseteq \Gamma$ und $b \notin X$,
2. τ hält für jede Eingabe $|^n$ an, $n \geq 0$ und $n \in N$,
3. $M = \{Res_\tau\,(|^n)\,|n \geq 0\}$

Für den Beweis von **Satz B** benötigen wir außerdem den Satz Q aus der Vorlesung vom 12.1.2001.

Satz Q:

Eine Sprache $L \subseteq T^\star$ kann genau dann von einer Grammatik erzeugt werden, wenn $\psi_L : T^\star \to T^\star$ Turing-berechenbar ist.

Satz B:

Es sei eine Menge $M \subseteq X^\star$ gegeben. Dann sind folgende Aussagen äquivalent.

(i) *M* ist aufzählbar.

(ii) *M* ist Ergebnisbereich einer Turing-Maschine.

(iii) *M* ist Haltebereich einer Turing-Maschine.

(iv) ψ_M ist partiell-rekursiv.

(v) *M* ist Element von *Chomsky-0*.

Vorgehensweise

Verwendung des sogenannten Ringschlusses $(i) \Leftrightarrow (ii) \Leftrightarrow (v) \Leftrightarrow (iv) \Leftrightarrow (iii) \Leftrightarrow (i)$ als Beweistechnik.

Beweis

$(i) \Rightarrow (ii)$**:**

Nach der Definition A wird der Fall, daß die Menge *M* leer ist $(M = \emptyset)$ von einer nichthaltenden Turing-Maschine berechnet.

$(ii) \Rightarrow (v)$**:**

Nach **Satz Q** kann zu jeder Turing-Maschine $\tau = (S, X, \Gamma, \delta, s_0, b)$ eine Grammatik $G_\tau = (N, T, P, \sigma)$ konstruiert werden mit $L(G_\tau) = \{w \# v \,|\, Res_\tau(w) = v\}$.
Ergänze G_τ zu $G_{\tau'}$, sodaß $L(G_{\tau'}) = \{v \,|\, Res_\tau(w) = v\}$. Damit ist $G_{\tau'}$ eine *Chomsky-0*-Grammatik.

$(v) \Rightarrow (iv)$**:**

Auch hier ist wieder der **Satz Q** anwendbar. Eine Funktion heißt *Turing-berechenbar*, wenn eine Turing-Maschine existiert, welche *L* akzeptiert. Wenn also eine Turing-Maschine existiert, welche *L* nach $\psi_L : T^\star \to T^\star$ berechnet und anhält ist *L* vom Typ *Chomsky-0*.

$(iv) \Rightarrow (iii)$**:**

Nach Definition von ψ_M gibt es eine Turing-Maschine, die für alle Wörter $w \in M$ mit Ausgabe $tt \in X^\star$ anhält. $\psi_M(w) = \begin{cases} tt, w \in M \\ \perp, \text{ sonst} \end{cases}$

$(iii) \Rightarrow (i)$**:**

Sei $\tau = (S, X, \Gamma, \delta, s_0, b)$ eine Turing-Maschine und $M \subseteq X^\star$ der Haltebereich dieser Turing-Maschine:

$M = \{v \in X^\star \,|Res_\tau(v) \text{ existiert}\}$.

Falls $M \neq \emptyset$, dann ist M aufzählbar. Nehmen wir also an, es sei also $M \neq \emptyset$, dann konstruieren wir einen Algorithmus , der zur Eingabe $|^n$ mit $n \geq 0$ ein Element $w_n \in M$ liefert, so daß $M = \{w_1, w_2, w_3, ...\}$ existiert.

Algorithmus

1. Da $M \neq \emptyset$, wähle ein Wort $z \in M$ beliebig fest.

2. Sei $a_1 \in X$ der erste Buchstabe (bzgl. der lexikographischen Reihenfolge).
   (1) $a_1 < a_2 < a_3 < ... \,|a_i \in X$
   (2) $u, v \in X^\star : \; u < v \Leftrightarrow$ entweder $|u| < |v|$oder falls $|u| = |v|$, $u \neq v$, $u = u_1u_2...u_n$, $v = v_1v_2...v_n$ und es gilt: $u_1 = v_1, ..., u_i = v_i$, $u_{i+1} \neq v_{i+1}$, dann ist $u_{i+1} < v_{i+1}$. Fortgesetzt auf $\leq$.

3. $N : X^\star \to X^\star$ Nachfolgerfunktion bezüglich der lexikongraphien Ordnung $(N(\varepsilon) = a_1)$, welche durch eine Turingmaschine realisiert werden kann.

4. Wir betrachten den Algorithmus, welcher zu $|^n$, $n \geq 0$ ein Wort $w_n \in M$ liefert. Die zugehörige Turing-Maschine $\hat{\tau}$ arbeitet so:

Anfang:

$|^n$ stehe auf dem Band, dann schreibe:

$$\#z\#\varepsilon\beta \notin s_0b\$\#a\#$$

dahinter

$$|^n\#z\#\varepsilon\beta \notin s_ob\$\#a\#$$

wobei $\varepsilon = w_0$ und wiederum daraus

$$|^n\#z\#\beta \notin s_ob\$\#a\#$$

Iteration:
Sei τ die Turing-Maschine mit Haltebereich *M*. Wenn τ eine Zwischensituation erreicht hat, dann entsteht folgender Zustand:

$$|^m\#z\#w_1\beta \notin u_1q_1v_1\$w_2\beta \notin u_2q_2v_2\$...\$w_s\beta \notin u_sq_sv_s\$\#w\#$$

wobei $u_j q_j v_j$ Konfigurationen von der Turing-Maschine τ sind. Dann überführt τ jede Komponente $u_j q_j v_j$ in die Folgekonfiguration $u'_j q'_j v'_j$ mit $q_j = \varepsilon$, falls die Übergangsfunktion $\delta\left(q_j, \cdot, \text{ irgendwas}\right) = \left(q'_j, \cdot, h\right)$ eine Haltekonfiguration liefert. Am Ende des Bandes stellt man die Startkonfiguration für *w* ein („$\notin q_0 w\$$“) und läßt die Turing-Maschine „$\#N\left(w\right)\#$“ dahinter schreiben.

$$|^n \#z\#w_1\beta \notin u'_1 q'_1 v'_1 \$ w_2\beta ... w_s\beta \notin u'_s q'_s v'_s \$ w\beta \notin s_0 w\$\#N\left(w\right)\#$$

Danach soll die Turing-Maschine folgenden Algorithmus ausführen:

if $m = 0$ **then** „*lösche alles bis auf z*“; **stop**
else $m := m - 1$ („*Streiche ein '|'*“)
while es existiert $q'_j = \epsilon$ **do**
nimm das kleinste *j* mit $q'_j = \epsilon$;
if $m = 0$ **then** „*lösche alles bis auf* w_j“; **stop**
else „*lösche* $w_j\beta \notin u'_j v'_j\$$“; $m := m - 1$;
fi
od
„*Stelle LSK auf das erste* $\notin$“;
fi

Wenn kein **stop** erreicht wurde, starte die Iteration erneut. Mit der Turing-Maschine $\hat{\tau}$ kann nach der Churchscher These[1] simuliert werden. Dabei simuliert $\hat{\tau}$ schrittweise alle Konfigurationsübergänge für alle Eingaben aus $X^\star$.

Dies geschied, indem immer wenn τ ein β schreibt, vergleicht $\hat{\tau}$ ihre Eingabe mit dem gerade erzeugten Wort. Sind sie gleich, so akzeptiert $\hat{\tau}$, sonst simuliert $\hat{\tau}$ weiterhin τ.

Wenn $w \in M$ ist, sodaß die Komponente, welche mit *w* gestartet wurde, dafür sorgt, das τ irgendwann terminiert. Für ein ganz bestimmtes Symbol wird ein *w* ausgegeben, aber auch nur solche, für die τ terminiert. □

[1] Die Annahme, daß die intuitive Vorstellung äquivalent der Klasse der partiell-rekursiven Funktionen ist

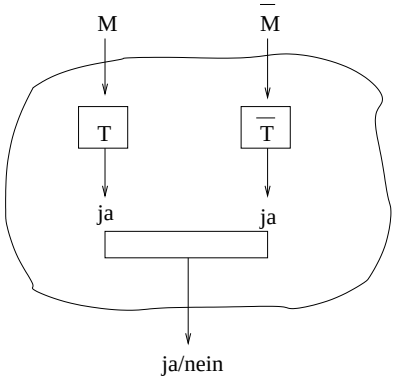
Unterschiede zwischen Aufzählbarkeit $\leftrightarrow$ Entscheidbarkeit

asymmetrische Phänomene	symmetrische Phänomene
aufzählbare Mengen *M* haltende *TM* Menge $\overline{M}$ nichthaltende *TM*	entscheidbare Mengen $\rightarrow x \in M$ und $\rightarrow x \notin M$

Der Komplex der Entscheidbarkeit ist sehr umfangreich und wurde in der Vorlesung nur angeschnitten.

Definition C

Es sei *X* ein endliches Alphabet. Eine Menge $M \subseteq X^\star$ heißt **entscheidbar**, wenn ihre charakteristische Funktion X_M berechbar ist.



Satz D

Eine Menge $M \subseteq X^\star$ ist entscheidbar, wenn *M* und $X^\star \setminus M$ aufzählbar sind.

Beweis

„⇒“: Die Menge *M* ist entscheidbar, wenn $\Rightarrow X_\Gamma$ berechenbar ist mit einer Turing-Maschine τ. Dazu wandelt man die Turing-Maschine τ so ab, sodaß τ in eine Endlosschleife gerät, wenn $X_M(w) = t$. Die neue Turing-Maschine berechnet τ_M.

„⇐“: **Beweisidee:** Man lasse eine Turing-Maschine τ_1 und eine Turing-Maschine τ_2 für M bzw. $X^\star \setminus M$ parallel Schritt für Schritt laufen. Die Maschine, welche terminiert, liefert die Antwort, ob ein Wort $W \in M$ oder $W \in X^\star \setminus M$ ist. □

Definition & Satz E

Es sei **ENT** die Klasse der entscheidbaren Sprachen.
Desweiteren **AUF** die Klasse der aufzählbaren Sprachen.
Dann gilt:

1. Jede Sprache ist *aufzählbar* und *entscheidbar.*
2. **ENT** ist abgeschlossen gegen Vereinigung $(\cup)$, Durchschnitt $(\cap)$, Komplement $(\overline{w})$, Sternbildung $(\star)$ und Konkatenation $(\cdot)$.
3. **AUF** ist abgeschlossen gegen Vereinigung $(\cup)$, Durchschnitt $(\cap)$, Sternbildung $(\star)$ und Konkatenation $(\cdot)$, aber nicht gegen das Komplement $(\overline{w})$.

Zum Abschluß noch ein paar einfache Beispiele zum Thema *Entscheibarkeitsprobleme*:

Das Wortproblem:

Liegt ein Wort *w* in der Sprache *L* oder nicht ? (Bekannt aus Theorie 1)

Das Leerheitsproblem:

Es sei eine Grammatik *G* gegeben. Existiert ein Akzeptor der $L(G) = \emptyset$ akzeptiert ?

Das Endlichheitsproblem:

Ist bei einer gegebenen Grammatik *G* $|L(G)| < \infty$?

Das Schnittproblem:

Ist der Schnitt zweier Grammatiken G_1 und G_2 leer ?

Das Äquivalenzproblem:

Definieren zwei reguläre Grammatiken G_1 und G_2 die selbe Sprache ?