Satz von Rice: Grundlagen, Beweis und Implikationen

René Filip[†]

***Abstract*—Der Satz von Rice hat in der Informatik weitreichende Konsequenzen, denn er sagt aus, dass es kein allgemeines Computerprogramm gibt, das überprüft, ob der eigene Code eine gewünschte Funktion berechnet oder nicht. Nachdem wir die Grundlagen und Definitionen geklärt haben, beweisen wir den Satz mittels "Many-to-One" Reduktion auf ein einfacheres Problem, für das sich ein direkter Beweis finden lässt.**

***Index Terms*—Satz von Rice, Theoretische Informatik, Entscheidbarkeit, Halteproblem.**

I. Einführung

II. Grundlagen und Definitionen

Definition 1 (Turingmaschine)**.** *Eine Turingmaschine (TM) ist definiert durch ein Septuple $(Q, \Gamma, B, \Sigma, \delta, q_0, F)$:*

1) *endliche Zustandsmenge Q*
2) *endliche Bandalphabet Γ*
3) *Leerzeichen $B \in \Gamma$*
4) *endliche Eingabealphabet $\Sigma \subseteq \Gamma \setminus B$*
5) *Zustandsübergangsfunktion*
   $\delta\colon Q \times \Gamma \to Q \times \Gamma \times (L, R, N)$
6) *Anfangszustand $q_0 \in Q$*
7) *Menge der akzeptierenden Endzustände $F \subseteq Q$*

Zwar kann eine Turingmaschine jedes Computerprogramm simulieren [1], doch kann man den Programmcode (sprich die Übergangsfunktion δ) nicht ändern. Ein Computer nach der Von-Neumann-Architektur (VNA) lässt sich stattdessen programmieren. Um den Satz von Rice und dessen Beweis sauber zu formulieren, müssen wir zunächst eine "universelle Turingmaschine" definieren:

Definition 2 (Universelle Turingmaschine)**.** ...

Die Universelle Turingmaschine nimmt nun als Eingabe nicht nur ein Wort $w \in \{0,1\}^*$, sondern auch ein Computerprogramm bzw. Gödelnummer $\langle M \rangle$. Die Gödelnummer ist eine Repräsentationsart einer Turingmaschine und ist wie folgt definiert:

Definition 3 (Gödelnummer)**.** *Sei M eine Turingmaschine $(\{q_1, \ldots, q_t\}, \{0, 1, B\}, B, \{0, 1\}, \delta, q_1, \{q_2\})$. Außerdem sei $X_1 = 0$, $X_2 = 1$, $X_3 = B$, $D_1 = L$, $D_2 = N$ und $D_3 = R$. Nun kann die z-te Zeile der Übergangsfunktions-Tabelle δ der Länge s wie folgt beschrieben werden:*

$$\delta(q_i, X_j) = (q_k, X_l, D_m) \Longrightarrow \mathrm{code}(z) := 0^i 1 0^j 1 0^k 1 0^l 1 0^m$$

wobei $i, k \geq 1$ und $j, l, m \in \{1, 2, 3\}$. Die Gödelnummer $\langle M \rangle$ der Turingmaschine M ist nun definiert als

$$\langle M \rangle := 111\,\mathrm{code}(1) 11\,\mathrm{code}(2) 11 \ldots 11\,\mathrm{code}(s) 111$$

Wir können leicht sehen, dass jedes Programm mit 111 anfängt und endet. Jede "Programmcode-Zeile" wird durch 11 separiert. $\mathrm{code}(z)$ enthält niemals den Teilstring 11. [2] beschreibt die Gödelnummer näher.

Definition 4 (Entscheidbarkeit)**.** *Eine Sprache L heißt rekursiv (oder entscheidbar), wenn es eine Turingmaschine gibt, die auf allen Eingaben stoppt und die Eingabe w genau dann akzeptiert, wenn $w \in L$ ist. [3]*

Eine Turingmaschine, die für jede Eingabe anhält, nennt man auch Entscheider.

III. Satz von Rice

...

Satz 5 (Satz von Rice)**.** *Sei $\mathcal{R}$ die Menge der von Turingmaschinen berechenbaren Funktionen und S eine Teilmenge von S mit $S \neq \emptyset$ und $S \neq \mathcal{R}$, das heißt S ist eine nicht triviale Teilmenge aller berechenbaren Funktionen. Dann ist die Sprache*

$$L(S) := \{\langle M \rangle \mid M \text{ berechnet eine Funktion aus } S\}$$

nicht rekursiv.

Für den Beweis von 5 siehe III-C.

...

A. Beweisstrategie

B. Aufbauende Sätze

Satz 6. *D ist nicht rekursiv.*

Beweis. ... □

Korollar 7. *$\overline{D}$ ist nicht rekursiv.*

Beweis. ... □

Satz 8. *H ist nicht rekursiv.*

Beweis. ... □

Satz 9. *H_ε ist nicht rekursiv.*

Beweis. ... □

C. Beweis des Satzes von Rice

Beweis. ... □

[†]DHBW Karlsruhe, TINF13B1, Seminar Theoretische Informatik 2016

REFERENCES

[1] I. Wegener, “Theoretische informatik - eine algorithmenorientierte einführung.” Berlin Heidelberg New York: Springer-Verlag, 2013, ch. 2, pp. 16 – 17.

[2] ——, “Theoretische informatik - eine algorithmenorientierte einführung.” Berlin Heidelberg New York: Springer-Verlag, 2013, ch. 2, p. 18.

[3] ——, “Theoretische informatik - eine algorithmenorientierte einführung.” Berlin Heidelberg New York: Springer-Verlag, 2013, ch. 2, p. 11.