

Churchsche These

Die Menge der Turing-berechenbaren Funktionen ist genau die Menge der im intuitiven Sinne berechenbaren Funktionen.

Varianten von Turing-Maschinen

- ▶ Turing-Maschinen mit mehreren Bändern
- ▶ Turing-Maschinen mit einseitig beschränktem Band
- ▶ Turing-Maschinen mit mehreren Leseköpfen
- ▶ Turing-Maschinen mit mehrdimensionalem Arbeitsband
- ▶ ...

Theorem

Turing-Maschinen sind genauso mächtig wie ihre Varianten.

Turingmaschinen mit mehreren Bändern

Turingmaschine mit k Bändern

$$TM = (S, I, A, d, s_0, F)$$

- $d(s, a_1, \dots, a_k) \subseteq S \times A^k \times \{l, r, n\}^k$ für alle $s \in S$ und alle $(a_1, \dots, a_k) \in A^k$,
- alle weiteren Komponenten wie bei (Einband-) Turingmaschinen.

Konfigurationen

Konfiguration: $u_1 s v_1, \dots, u_k s v_k$ mit $s \in S$, $u_j, v_j \in A^*$
($j = 1, \dots, k$)

Anfangskonfiguration: $\lambda s_0 w, s_0, \dots, s_0$ mit $w \in I^*$

Folgekonfiguration

Falls $(s', b_1, \dots, b_k, x_1, \dots, x_k) \in d(s, a_1, \dots, a_k)$:

$$u_1 s a_1 v_1, \dots, u_k s a_k v_k \vdash u'_1 s' v'_1, \dots, u'_k s' v'_k,$$

wobei man $u'_j s' v'_j$ aus $u_j s a_j v_j$ wie folgt erhält:

1. Überschreibe a_j durch b_j .
2. Ersetze s durch s' .
3. Verschiebe s' gemäß x_j .

Erkannte Sprache

Sei $TM = (S, I, A, d, s_0, F)$ eine Turingmaschine mit k Bändern.

Dann ist $L(TM)$ die von TM erkannte Sprache mit

$$L(TM) = \left\{ w \in I^* \mid s_0 w, s_0, \dots, s_0 \vdash^* \right. \\ \left. u_1 s v_1, \dots, u_k s v_k, s \in F \right\}$$

Äquivalenz zwischen Mehrband-Turingmaschinen und Turingmaschinen

Theorem

Mehrband-Turingmaschinen erkennen dieselben Sprachen wie (Einband)-Turingmaschinen.

Beweisskizze: Sei $TM(k)$ eine Turingmaschine mit $k > 1$ Bändern. Simuliere $TM(k)$ wie folgt:

1. Schreibe alle k Eingabewörter nebeneinander auf das Arbeitsband, so dass
 - sie durch besondere Symbole getrennt sind
 - die Position der Leseköpfe mittels spezieller Symbole kodiert wird.
2. Führe nacheinander an den markierten Stellen die Aktionen von $TM(k)$ aus. (Dafür muss evtl. durch Verschieben des Bandinhalts Platz geschaffen werden.)

Äquivalenz zwischen deterministischen und nichtdeterministischen Turingmaschinen

Deterministische Turingmaschine

TM ist deterministisch, falls $d(s, a)$ für jedes $s \in S$ und $a \in A$ höchstens ein Element enthält.

Theorem

Nichtdeterministische Turingmaschinen erkennen dieselben Sprachen wie deterministische Turingmaschinen.

Beweisskizze: Sei $TM = (S, I, A, d, s_0, F)$ eine nicht-deterministische Turingmaschine. Sei

$$k = \max \{ |d(s, a)| \mid s \in S, a \in A \}.$$

Simuliere TM durch eine deterministische Turingmaschine mit 3 Bändern:

1. Schreibe die Eingabe von TM auf das erste Band.
2. Erzeuge auf Band 2 nach und nach alle Wörter über $\{1, \dots, k\}$.
3. Kopiere für jedes Wort u auf Band 2 die Eingabe auf Band 3 und simuliere TM unter Benutzung der Zahlenfolge in u .

Turingmaschinen und Typ-0-Sprachen

Theorem

Die von Turingmaschinen erkannten Sprachen sind genau die Typ-0-Sprachen.

Übersetzung von Chomsky-Grammatiken in Turingmaschinen

Sei $G = (N, T, P, S)$ eine Chomsky-Grammatik. Dann arbeitet $TM(G)$ wie folgt:

1. Wähle eine Regel $u ::= v \in P$.
2. Ersetze ein beliebiges Vorkommen von v im aktuellen Bandinhalt w durch u , falls v in w vorkommt. (Ist $|u| \neq |v|$ müssen Teile von w verschoben werden.)
3. Ist S der aktuelle Bandinhalt, gehe in einen Endzustand; sonst gehe zu 1.

Korrektheit der Übersetzung

Theorem

$$L(G) = L(TM(G))$$

Übersetzung von Turingmaschinen in Typ-0-Grammatiken

Sei $TM = (S, I, A, d, s_0, F)$ eine Turingmaschine. Dann enthält $G(TM) = (N, I, P, S')$ die folgenden Regeln:

1. Regeln, die aus S' Wörter der Form

$$\$s_0w\#w \text{ mit } w \in I^*, \$, \# \notin A$$

erzeugen.

2. Regeln, die $\$s_0w\#w$ in $\$usv\#w$ umwandeln, falls

$$s_0w \xrightarrow{*} usv.$$

3. Regeln, die $\$usv\#w$ in w umwandeln, falls $s \in F$.

Definition der Regeln von $G(TM)$

1. Regeln für $S' \xrightarrow{*} \$s_0w\#w$: (vgl. Übung)

2. Regeln für $\$s_0w\#w \xrightarrow{*} \$usv\#w$:

- Falls $(b, s', l) \in d(s, a)$:

$$csa ::= s'cb \text{ für alle } c \in A$$

$$\$sa ::= \$s'\square b$$

- weitere Fälle analog.

3. Regeln für $\$usv\#w \xrightarrow{*} w$:

- $s ::= L, Lc ::= L, cL ::= L, \$L\# ::= \lambda$
(mit $s \in F, c \in A$)

Korrektheit der Übersetzung

Theorem

$$L(TM) = L(G(TM))$$

Linear beschränkte Turingmaschine

- ▶ Benutzt nur den Platz, auf dem die Eingabe steht.
- ▶ Erkennt Randfelder durch spezielle Symbole.
- ▶ Markierung des linken Randfelds gleich nach dem Start.
- ▶ Eingabealphabet: $I \cup \hat{I}$ mit $\hat{I} = \{\hat{x} \mid x \in I\}$.
- ▶ Startkonfigurationen haben die Form: $s_0 w \hat{x}$ mit $w \in I^*$ und $\hat{x} \in \hat{I}$.
- ▶ λ kann nicht erkannt werden.

Definition

Linear beschränkte Turingmaschine

Eine Turingmaschine $TM = (S, I, A, d, s_0, F)$ heißt **linear beschränkt**, falls für alle $w \in I^*$, $u, v \in A^*$, $s \in S$ gilt:

$$s_0w \stackrel{*}{\vdash} usv \implies |w| = |uv|.$$

Erkannte Sprache

Die von einer linear beschränkten Turingmaschine $TM = (S, I, A, d, s_0, F)$ erkannte Sprache ist definiert durch

$$\{wx \mid w \in I^*, x \in I, s_0 w \hat{x} \xrightarrow{*} usv, s \in F\}.$$

Theorem

Die von linear beschränkten Turingmaschinen erkannten Sprachen sind genau die monotonen Sprachen.

Offenes Problem

Können deterministische linear beschränkte Turingmaschinen dieselben Sprachen erkennen wie nichtdeterministische linear beschränkte Turingmaschinen?

Unlösbarkeit des Wortproblems für Typ-0-Sprachen

Theorem

Das Wortproblem für Typ-0-Sprachen ist nicht berechenbar (z.B. durch eine CE-S-Spezifikation, eine Turing-Maschine oder ein WHILE-Programm).

Beweis der Unlösbarkeit mittels Reduktion

Reduktion

Seien $dp_i: A_i \rightarrow \text{BOOL}$ ($i = 1, 2$) Entscheidungsprobleme.

dp_1 ist auf dp_2 **reduzierbar**, falls es eine berechenbare Funktion $red: A_1^* \rightarrow A_2^*$ gibt, so dass

$$dp_1(w) = T \iff dp_2(red(w)) = T \text{ für alle } w \in A_1^*.$$

Beweis der Unlösbarkeit mittels Reduktion

Theorem

dp ist nicht berechenbar.

Beweis

Wähle ein anderes Entscheidungsproblem dp_0 , dessen Unlösbarkeit bereits bekannt ist.

Reduziere dp_0 auf dp .