

Grundlagen der Theoretischen Informatik

Turingmaschinen und rekursiv aufzählbare Sprachen (V)

15.07.2015

Viorica Sofronie-Stokkermans

e-mail: sofronie@uni-koblenz.de

Übersicht

1. Motivation
2. Terminologie
3. Endliche Automaten und reguläre Sprachen
4. Kellerautomaten und kontextfreie Sprachen
5. Turingmaschinen und rekursiv aufzählbare Sprachen
6. Berechenbarkeit, (Un-)Entscheidbarkeit
7. Komplexitätsklassen P und NP

Übersicht

- Determinierte Turing-Maschinen (DTMs)
- Varianten von Turing-Maschinen
- Indeterminierte Turing-Maschinen (NTMs)
- **Universelle determinierte Turing-Maschinen**
- Entscheidbar/Aufzählbar
- Determinierte Turing-Maschinen entsprechen Typ 0
- Unentscheidbarkeit

Universelle determinierte Turing-Maschinen

Vergleich Turing-Maschine / „normaler“ Computer

Turing-Maschinen sind sehr mächtig.

Wie mächtig sind sie wirklich?

- Eine Turing-Maschine hat ein vorgegebenes „Programm“ (Regelmenge)
- „Normale“ Computer können beliebige Programme ausführen.

Tatsächlich geht das mit Turing-Maschinen auch!

Universelle determinierte Turing-Maschinen

Turing-Maschine, die andere TMen simuliert

- Universelle TM \mathcal{U} bekommt als Eingabe:
 - die Regelmenge einer beliebigen Turing-Maschine \mathcal{M} und
 - ein Wort w , auf dem \mathcal{M} rechnen soll.
- \mathcal{U} simuliert \mathcal{M} , indem sie jeweils nachschlägt, welchen δ -Übergang \mathcal{M} machen würde.

Universelle determinierte Turing-Maschinen

TM als Eingabe für eine andere TM

Frage:

In welches Format fasst man die Regeln einer DTM \mathcal{M} am besten, um sie einer universellen DTM als Eingabe zu geben?

Was muss man angeben, um eine DTM komplett zu beschreiben?

- das Alphabet,
- die Zustände,
- die δ -Übergänge
- den Startzustand.

Universelle determinierte Turing-Maschinen

Standardisierung von Alphabet, Zustandsmenge, Startzustand

- Unendliches Alphabet $\Sigma_\infty = \{a_0, a_1, \dots\}$,
so dass das Alphabet jeder DTM eine Teilmenge von Σ_∞ ist.
- Namen der Zustände einer DTM sind egal.
Sie seien also q_1, \dots, q_n
(n kann dabei von DTM zu DTM verschieden sein).
- Sei q_1 immer der Startzustand, und
bezeichne q_0 den Haltezustand

Damit:

**Wir können eine DTM komplett beschreiben,
indem wir nur ihre δ -Übergänge beschreiben.**

Universelle determinierte Turing-Maschinen

(Mögliche) Kodierung der Übergangsrelation Die DTM $\mathcal{L}_\#$ habe die Regeln

$$\begin{aligned}
 q_1, \# &\mapsto q_2, L & q_2, \# &\mapsto h, \# \\
 q_1, | &\mapsto q_2, L & q_2, | &\mapsto q_2, L
 \end{aligned}$$

Dabei sei: $\# = a_0$ und $| = a_1$

Dann kann die DTM $\mathcal{L}_\#$ so beschrieben werden:

	a_0	a_1
q_1	q_2, L	q_2, L
q_2	h, a_0	q_2, L

oder kürzer:

$$\begin{aligned}
 Z \ S \ 2\lambda \ S \ 2\lambda \\
 Z \ S \ 00 \ S \ 2\lambda
 \end{aligned}$$

Universelle determinierte Turing-Maschinen

(Mögliche) Kodierung der Übergangsrelation Dabei steht:

- Z für “nächste Zeile”
- S für “nächste Spalte”
- λ für “links”, ρ für “rechts”
- die Zahl n für den n -ten Zustand und für das n -te Zeichen von Σ_∞ .

Damit ist die DTM insgesamt durch ein einziges Wort beschrieben:

$ZS2\lambda S2\lambda ZS00S2\lambda$

Universelle determinierte Turing-Maschinen

Gödelisierung

Ein Verfahren, jeder Turing-Maschine eine Zahl oder ein Wort (**Gödelzahl** bzw. **Gödelwort**) so zuzuordnen, dass man aus der Zahl bzw. dem Wort die Turing-Maschine effektiv rekonstruieren kann.

Kurt Gödel

Kurt Gödel 1906-1978

- Bedeutendster Logiker des 20. Jahrhunderts
- Vollständigkeitssatz (1929)
Promotion in Wien
- Unvollständigkeitssatz (1931)
Idee der Gödelisierung
- Beweis der Unabhängigkeit der
Kontinuumshypothese
- Dozent in Princeton,
befreundet mit Albert Einstein
- Tragischer Tod:
Verfolgungswahn, Depressionen,
Tod durch Unterernährung.



Übersicht

- Determinierte Turing-Maschinen (DTMs)
- Varianten von Turing-Maschinen
- Indeterminierte Turing-Maschinen (NTMs)
- Universelle determinierte Turing-Maschinen
- **Entscheidbar/Aufzählbar**
- Determinierte Turing-Maschinen entsprechen Typ 0
- Unentscheidbarkeit

Entscheidbar/Aufzählbar

Akzeptieren

Eine DTM **akzeptiert** eine Sprache L , wenn sie

- für jedes Eingabe-Wort $w \in L$ irgendwann hält
- für jedes Wort $v \notin L$ unendlich lang rechnet oder hängt

Entscheiden

Eine DTM **entscheidet** eine Sprache L , wenn sie

- für jedes Eingabe-Wort $w \in L$ hält mit dem Bandinhalt Y ("Yes")
- für jedes Wort $v \notin L$ hält mit dem Bandinhalt N ("No")

Akzeptierbarkeit und Entscheidbarkeit

Definition [Entscheidbar]

L sei eine Sprache über Σ_0 mit $\#, N, Y \notin \Sigma_0$.

$M = (K, \Sigma, \delta, s)$ sei eine DTM mit $\Sigma_0 \subseteq \Sigma$.

M **entscheidet** L , falls für alle $w \in \Sigma_0^*$ gilt:

$$s, \# \underline{w} \# \vdash_M^* \begin{cases} h, \# \underline{Y} \# & \text{falls } w \in L \\ h, \# \underline{N} \# & \text{sonst} \end{cases}$$

L heißt **entscheidbar**, falls es eine DTM gibt, die L entscheidet.

Akzeptierbarkeit und Entscheidbarkeit

Definition [Akzeptierbar]

L sei eine Sprache über Σ_0 mit $\#, N, Y \notin \Sigma_0$.

$M = (K, \Sigma, \delta, s)$ sei eine DTM mit $\Sigma_0 \subseteq \Sigma$.

\mathcal{M} **akzeptiert** ein Wort $w \in \Sigma_0^*$, falls \mathcal{M} bei Input w hält.

\mathcal{M} **akzeptiert die Sprache L** , falls für alle $w \in \Sigma_0^*$ gilt:

\mathcal{M} akzeptiert w *genau dann wenn* $w \in L$

L heißt **akzeptierbar** (oder auch **semi-entscheidbar**), falls es eine DTM gibt, die L akzeptiert.

Rekursiv aufzählbar

Definition [Rekursiv Aufzählbar (recursively enumerable)]

L sei eine Sprache über Σ_0 mit $\#, N, Y \notin \Sigma_0$.

$M = (K, \Sigma, \delta, s)$ sei eine DTM mit $\Sigma_0 \subseteq \Sigma$.

\mathcal{M} **zählt** L **auf**, falls es einen Zustand $q_B \in K$ gibt (den **Blinkzustand**), so dass:

$$L = \{w \in \Sigma_0^* \mid \exists u \in \Sigma^* : s, \underline{\#} \vdash_{\mathcal{M}}^* q_B, \#w\underline{\#}u\}$$

L heißt **rekursiv aufzählbar**, falls es eine DTM gibt, die L aufzählt.

Rekursiv aufzählbar

Achtung: aufzählbar \neq abzählbar.

Unterschied

M **abzählbar:** Es gibt eine surjektive Abbildung der natürlichen Zahlen auf M

M **aufzählbar:** Diese Abbildung kann von einer Turing-Maschine berechnet werden.

Wegen Endlichkeit der Wörter und des Alphabets sind alle Sprachen abzählbar.

Aber nicht alle Sprachen sind aufzählbar.

Aufzählbarkeit / Entscheidbarkeit / Unentscheidbarkeit

Fragen:

Gibt es nicht-aufzählbare Probleme?

Gibt es nicht-entscheidbare Probleme?

Aufzählbarkeit / Entscheidbarkeit / Unentscheidbarkeit

Fragen:

Gibt es nicht-aufzählbare Probleme?

Gibt es nicht-entscheidbare Probleme?

Antwort: Ja, es gibt nicht-aufzählbare Probleme und es gibt nicht-entscheidbare Probleme.

Beweis: Die Menge aller Sprachen ist nicht abzählbar

Die Turingmaschinen können abgezählt werden: M_1, M_2, \dots

Die Mächtigkeit der Menge aller Sprachen ist größer als die Mächtigkeit der Menge aller Turingmaschinen.

⇒ es gibt Sprachen, für denen es keine TM gibt, die sie aufzählt.

⇒ es gibt Sprachen, für denen es keine TM gibt, die sie entscheidet.

Rekursiv aufzählbar: Beispiele

Rekursiv aufzählbar aber nicht entscheidbar

Folgende Mengen sind rekursiv aufzählbar aber *nicht* entscheidbar:

- Die Menge der Gödelisierungen aller haltenden Turing-Maschinen
- Die Menge aller terminierenden Programme
- Die Menge aller allgemeingültigen prädikatenlogischen Formeln

Akzeptierbar = Rekursiv aufzählbar

Theorem [Akzeptierbar = Rekursiv Aufzählbar].

Eine Sprache ist genau dann rekursiv aufzählbar, wenn sie akzeptierbar ist.

Akzeptierbar = Rekursiv aufzählbar

Theorem [Akzeptierbar = Rekursiv Aufzählbar].

Eine Sprache ist genau dann rekursiv aufzählbar, wenn sie akzeptierbar ist.

Beweis “ \Rightarrow ”

Sei L rekursiv aufzählbar

Es gibt also eine DTM \mathcal{M}_L , die L aufzählt.

Zur Erinnerung:

\mathcal{M}_L **zählt** L **auf**, falls es einen Zustand $q_B \in K$ gibt (den **Blinkzustand**), so dass:

$$L = \{w \in \Sigma_0^* \mid \exists u \in \Sigma^* : s, \underline{\#} \vdash_{\mathcal{M}}^* q_B, \#w\underline{\#}u\}$$

Zu zeigen: L ist akzeptierbar, d.h. es existiert eine DTM, die L akzeptiert.

Akzeptierbar = Rekursiv aufzählbar

Beweis (Fortsetzung)

Wir konstruieren aus \mathcal{M}_L eine 2-DTM \mathcal{M} , die L akzeptiert:

- \mathcal{M} wird gestartet mit

$$s_0, \quad \#w\underline{\#}$$
$$\underline{\#}$$

- \mathcal{M} simuliert auf Band 2 die Maschine \mathcal{M}_L .
- Wenn \mathcal{M}_L den **Blinkzustand** q_B erreicht, dann enthält Band 2 von \mathcal{M} ein Wort

$$\#w'\underline{\#}u$$

mit $w' \in L$.

Akzeptierbar = Rekursiv aufzählbar

Beweis (Fortsetzung)

- Nach Erreichen des Blinkzustands:
 \mathcal{M} vergleicht w und w' .
 - Falls $w = w'$, dann hält \mathcal{M} : $w \in L$.
 - Ansonsten simuliert \mathcal{M} auf Band 2 weiter die Arbeit von \mathcal{M}_L .
- Wenn \mathcal{M}_L hält, ohne das Wort w auf Band 2 erzeugt zu haben, gerät \mathcal{M} in eine Endlosschleife.

Akzeptierbar = Rekursiv aufzählbar

Beweis (Fortsetzung)

“ \Leftarrow ”

Sei L akzeptierbar

Es gebe also eine DTM \mathcal{M}_L , die L akzeptiert.

Zu zeigen: L ist rekursiv aufzählbar.

Akzeptierbar = Rekursiv aufzählbar

Beweis (Fortsetzung)

“ \Leftarrow ”

Sei L akzeptierbar

Es gebe also eine DTM \mathcal{M}_L , die L akzeptiert.

Zu zeigen: L ist rekursiv aufzählbar.

Wir konstruieren eine DTM \mathcal{M} , die L rekursiv aufzählt.

Grundidee:

- die Wörter aus Σ^* der Reihe nach aufzählen
- jedes Wort der Maschine \mathcal{M}_L vorlegen
- wenn \mathcal{M}_L das Wort akzeptiert, in den Blinkzustand q_B gehen

Akzeptierbar = Rekursiv aufzählbar

Beweis (Fortsetzung)

Problem: \mathcal{M}_L akzeptiert, sie entscheidet nicht.

Wenn \mathcal{M}_L ein Wort nicht akzeptiert, rechnet sie unendlich.

Lösung: Wir betrachten die Rechnung von \mathcal{M}_L zu allen Wörtern aus Σ^*
gleichzeitig.

Akzeptierbar = Rekursiv aufzählbar

Beweis (Fortsetzung)

$\Sigma^* = \{w_1, w_2, w_3, \dots\}$ (in lexikalischer Reihenfolge aufgezählt).

Dann rechnet \mathcal{M} so:

- Im ersten Durchlauf berechne den ersten Rechenschritt von \mathcal{M}_L für w_1 .
- Im zweiten Durchlauf berechne
 - die ersten zwei Rechenschritte von \mathcal{M}_L für w_1 ,
 - einen Rechenschritt für w_2 .
- Im dritten Durchlauf berechne drei Rechenschritte für w_1 , zwei für w_2 und einen für w_3 und so weiter.

Immer wieder bei der Startkonfiguration anfangen: So müssen wir uns den Bandinhalt der Rechnung von \mathcal{M}_L zu w_i nach j Schritten nicht merken.

Akzeptierbar = Rekursiv aufzählbar

Beweis (Fortsetzung)

Wenn \mathcal{M} so rechnet, dann gilt:

- \mathcal{M} fängt für jedes $w_i \in \Sigma^*$ in endlicher Zeit (nämlich im i -ten Durchlauf) an, die Arbeit von \mathcal{M}_L zu w_i zu simulieren, und
- falls $w_i \in L$ und falls \mathcal{M}_L , gestartet mit $s, \#w_i\#$, in j Schritten einen Haltezustand erreicht, dann erreicht \mathcal{M} nach endlicher Zeit (nämlich im $i + j$ -ten Durchlauf) den Haltezustand von \mathcal{M}_L in der Rechnung zu w_i .

Akzeptierbar = Rekursiv aufzählbar

Beweis (Ende)

- Wenn \mathcal{M} bei Simulation von \mathcal{M}_L zur Eingaben w_i auf eine Haltekonfiguration trifft, dann ist $w_i \in L$.
- \mathcal{M} nimmt dann eine Konfiguration

$$q_B, \#w_i\#u_i$$

ein – q_B ist der Blinkzustand.

- In der Nebenrechnung u_i steht, welche Teilrechnung von \mathcal{M}_L als nächste zu simulieren ist.

Also zählt \mathcal{M} die $w \in \Sigma^*$ auf, für die \mathcal{M}_L hält, und das sind gerade die $w \in L$. \square

Entscheidbarkeit und Akzeptierbarkeit

Theorem. Jede entscheidbare Sprache ist akzeptierbar.

Beweis

Sei L eine entscheidbare Sprache und \mathcal{M} eine DTM, die L entscheidet.

Dann wird L akzeptiert von der DTM \mathcal{M}' ,
die zunächst \mathcal{M} simuliert und danach in eine Endlosschleife geht, falls \mathcal{M}
mit $h, \# \underline{N} \#$ endet.

Entscheidbarkeit und Akzeptierbarkeit

Theorem [Komplement einer entscheidbaren Sprache ist entscheidbar]

Das Komplement einer entscheidbaren Sprache ist entscheidbar.

Entscheidbarkeit und Akzeptierbarkeit

Theorem [Komplement einer entscheidbaren Sprache ist entscheidbar]

Das Komplement einer entscheidbaren Sprache ist entscheidbar.

Beweis

Sei L eine entscheidbare Sprache und \mathcal{M} eine DTM, die L entscheidet.

Dann wird \bar{L} entschieden von einer DTM \mathcal{M}' ,
die genau wie \mathcal{M} rechnet
und nur am Schluß die Antworten Y und N vertauscht. \square

Entscheidbarkeit und Akzeptierbarkeit

Theorem [Charakterisierung von Entscheidbarkeit]

Eine Sprache L ist genau dann entscheidbar, wenn sie und ihr Komplement akzeptierbar sind.

Entscheidbarkeit und Akzeptierbarkeit

Theorem [Charakterisierung von Entscheidbarkeit]

Eine Sprache L ist genau dann entscheidbar, wenn sie und ihr Komplement akzeptierbar sind.

Beweis

“ \Rightarrow ”

Sei L ist entscheidbar.

Zu zeigen: L und \bar{L} sind akzeptierbar.

- L ist entscheidbar, also ist L akzeptierbar
- L ist entscheidbar, also ist \bar{L} entscheidbar
- \bar{L} ist entscheidbar, also ist \bar{L} akzeptierbar

Entscheidbarkeit und Akzeptierbarkeit

Beweis (Fortsetzung)

“ \Leftarrow ”

Seien L und \bar{L} akzeptierbar.

Zu zeigen: L ist entscheidbar.

- Sei \mathcal{M}_1 eine DTM, die L akzeptiert.
- Sei \mathcal{M}_2 eine DTM, die \bar{L} akzeptiert.

Daraus konstruieren wir eine 2-DTM \mathcal{M} , die L entscheidet:

- \mathcal{M} wird gestartet mit

$s_0, \# w \underline{\#}$

$\underline{\#}$

- \mathcal{M} kopiert w auf Band 2.

Entscheidbarkeit und Akzeptierbarkeit

Beweis (Ende)

- M simuliert abwechselnd
 - einen Schritt von \mathcal{M}_1 auf Band 1 und
 - einen Schritt von \mathcal{M}_2 auf Band 2.
- Das tut \mathcal{M} , bis entweder \mathcal{M}_1 oder \mathcal{M}_2 hält.
- Eine von beiden muss halten: w gehört entweder zu L oder zu \bar{L} .
- Wenn \mathcal{M}_1 hält, dann hält \mathcal{M} mit
 - $\#Y\#$ auf Band 1 und
 - $\#$ auf Band 2.
- Wenn \mathcal{M}_2 hält, dann hält \mathcal{M} mit
 - $\#N\#$ auf Band 1 und
 - $\#$ auf Band 2. \square

Übersicht

- Determinierte Turing-Maschinen (DTMs)
- Varianten von Turing-Maschinen
- Indeterminierte Turing-Maschinen (NTMs)
- Universelle determinierte Turing-Maschinen
- Entscheidbar/Aufzählbar
- **Determinierte Turing-Maschinen entsprechen Typ 0**
- Unentscheidbarkeit

Rekursiv Aufzählbar = Typ 0

Zur Erinnerung

Formale Sprachen sind vom **Typ 0**, wenn sie durch beliebige Grammatiken (keinerlei Einschränkungen) erzeugt werden können.

Rekursiv Aufzählbar = Typ 0

Theorem [Rekursiv aufzählbar = Typ 0].

Die rekursiv aufzählbaren Sprachen

(also die durch DTMn akzeptierbaren Sprachen)

sind genau die durch beliebige Grammatiken erzeugten Sprachen

(also die vom Typ 0).

Rekursiv Aufzählbar = Typ 0

Theorem [Rekursiv aufzählbar = Typ 0].

Die rekursiv aufzählbaren Sprachen

(also die durch DTMn akzeptierbaren Sprachen)

sind genau die durch beliebige Grammatiken erzeugten Sprachen

(also die vom Typ 0).

Beweisidee

- Zu jeder Turing-Maschine kann eine Grammatik konstruiert werden, deren Ableitungsschritte die Rechenschritte der TM simulieren (spezielle Variable markiert Position des Schreib-/Lesekopfes).
- Zu jeder Grammatik kann eine indeterminierte Turing-Maschine (und damit auch eine DTM) konstruiert werden, deren Rechenschritte den Ableitungsschritten der Grammatik entsprechen.

Beweisdetails: Buch von Erk und Priese, Seiten 198-201.

Übersicht

- Determinierte Turing-Maschinen (DTMs)
- Varianten von Turing-Maschinen
- Indeterminierte Turing-Maschinen (NTMs)
- Universelle determinierte Turing-Maschinen
- Entscheidbar/Aufzählbar
- Determinierte Turing-Maschinen entsprechen Typ 0
- **Unentscheidbarkeit**

Zentrale Fragestellung

Welche Funktionen sind durch einen Algorithmus berechenbar?

bzw.

Welche Problemen sind durch einen Algorithmus entscheidbar?

Die Motivation, die Entscheidbarkeit und Unentscheidbarkeit zu studieren, stammt ursprünglich von dem Mathematiker David Hilbert:

Anfang des 20. Jahrhunderts formulierte er einen Forschungsplan, dessen Ziel die Entwicklung eines Formalismus was, in dem man alle mathematischen Probleme lösen konnte.



Zentrale Fragestellung

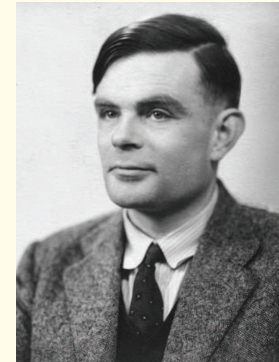
Welche Funktionen sind durch einen Computer berechenbar?

bzw.

Welche Probleme kann ein Computer entscheiden?

Um diese Fragen in einem mathematisch exakten Sinne klären zu können, müssen wir klären was eigentlich ein Computer ist.

Rechnermodell: [Turingmaschinen](#)



Alan Turing

Church-Turing These

Es wurde festgestellt, dass alle vernünftigen Modelle zur Spezifikation der algorithmischen Lösbarkeit äquivalent zu Turingmaschinen sind.

mehr darüber: Vertiefung Theoretische Informatik

Dies führte zu der Formulierung der sogenannten Church-Turing These:

Church-Turing These

Die Klasse der Turing berechenbaren Funktionen stimmt mit der Klasse der “intuitiv berechenbaren” Funktionen überein.

