

Grundlagen der Theoretischen Informatik

Turingmaschinen und rekursiv aufzählbare Sprachen (V)

6.07.2017

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

Indeterminierte Turing-Maschinen (NTMs)

Definition (Indeterminierte Turing-Maschine, NTM)

Eine **indeterminierte Turing-Maschine** \mathcal{M} ist ein Tupel

$$M = (K, \Sigma, \Delta, s)$$

Dabei sind K, Σ, s definiert wie bei determinierten Turing-Maschinen.

Übergangsrelation:

$$\Delta \subseteq (K \times \Sigma) \times ((K \cup \{h\}) \times (\Sigma \cup \{L, R\}))$$

Indeterminierte Turing-Maschinen (NTMs)

Mehrere Nachfolgekonfigurationen

Konfigurationen sind definiert wie bei DTMs.

Nun kann eine Konfiguration aber mehrere mögliche Nachfolgekonfigurationen haben.

Definition (NTM: Halten, Hängen, Akzeptieren) Sei $\mathcal{M} = (K, \Sigma, \Delta, s_0)$ eine indeterminierte Turing-Maschine.

- \mathcal{M} **hält** bei Input w , falls es **unter den möglichen Rechnungen**, die \mathcal{M} wählen kann, **eine gibt**, so dass \mathcal{M} eine Haltekonfiguration erreicht.
- \mathcal{M} **hängt** in einer Konfiguration, wenn es keine (durch Δ definierte) Nachfolgekonfiguration gibt.
- \mathcal{M} **akzeptiert** ein Wort w , falls sie **von $s, \#w\#$ aus einen Haltezustand erreichen kann**, und \mathcal{M} akzeptiert eine Sprache L , wenn sie genau alle Wörter $w \in L$ akzeptiert.

Indeterminierte Turing-Maschinen (NTMs)

Theorem [Simulation von NTM durch DTM]

Jede Sprache, die von einer indeterminierten Turing-Maschine akzeptiert wird, wird auch von einer Standard-DTM akzeptiert.

Beweis (Anfang) Sei

- L eine Sprache über Σ_0^* mit $\# \notin \Sigma_0$;
- $\mathcal{M} = (K, \Sigma, \Delta, s)$ eine indeterminierte Turing-Maschine, die L akzeptiert.

Indeterminierte Turing-Maschinen (NTMs)

Beweis (Fortsetzung) Wir konstruieren zu \mathcal{M} eine Standard-DTM \mathcal{M}' , die so rechnet:

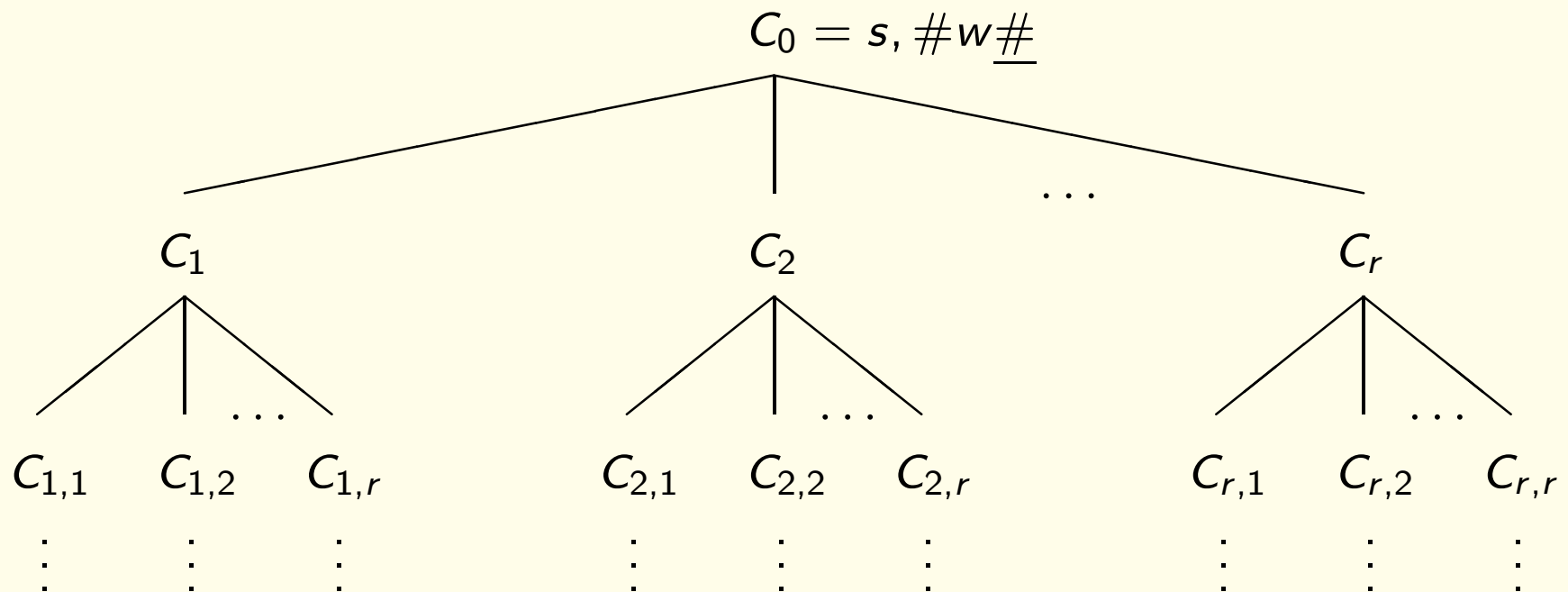
- \mathcal{M}' durchläuft systematisch **alle** Rechnungen von \mathcal{M} , und sucht dabei nach einer Haltekonfiguration.
- \mathcal{M}' hält genau dann, wenn sie eine Haltekonfiguration von \mathcal{M} findet.

Indeterminierte Turing-Maschinen (NTMs)

Beweis (Fortsetzung) Suchbaum, Rechnungsbaum:

Stelle alle Rechnungen von \mathcal{M} von einer Startkonfiguration C_0 dar als einen Baum mit Wurzel C_0 .

Ein Ast ist eine mögliche Rechnung von \mathcal{M} .



Indeterminierte Turing-Maschinen (NTMs)

Beweis (Fortsetzung)

Problem:

Es kann zur Startkonfiguration

$$C_0 = s, \#w\underline{\#}$$

unendlich viele Rechnungen von \mathcal{M} geben,
und jede einzelne von ihnen kann unendlich lang sein.

Wir können also nicht erst einen Ast ganz durchlaufen und dann den nächsten Ast durchsuchen.

Indeterminierte Turing-Maschinen (NTMs)

Beweis (Fortsetzung)

Die Lösung:

Breitensuche

Durchlaufe den Rechnungsbaum nicht depth-first, sondern per **iterative deepening**.

- Untersuche alle möglichen Rechnungen bis zum ersten Schritt.
- Untersuche alle möglichen Rechnungen bis zum zweiten Schritt.
- Untersuche alle möglichen Rechnungen bis zum dritten Schritt.
- usw.

Indeterminierte Turing-Maschinen (NTMs)

Beweis (Fortsetzung)

Können wir damit denn in endlicher Zeit eine Haltekonfiguration finden, falls es eine gibt?

Problem:

Kann der Rechnungsbaum nicht nur **unendlich tief**, sondern auch **unendlich breit** werden?

Nein, denn:

Maximale Anzahl von Nachfolgekonfigurationen

$$r = \max\{|\Delta(q, a)| \mid q \in K, a \in \Sigma\}$$

Indeterminierte Turing-Maschinen (NTMs)

Beweis (Fortsetzung)

\mathcal{M}' kann (z.B.) als eine 3-DTM gewählt werden:

- **Auf dem ersten Band steht immer das Eingabewort w .**

Da die Rechnung immer wieder neu mit $s, \#w\#$ von \mathcal{M} beginnt, wird das Eingabewort immer wieder gebraucht.

- **Auf dem zweiten Band steht, welcher Weg durch den Rechnungsbaum gerade verfolgt wird.**

Der Einfachheit halber: Wenn eine Konfiguration weniger als r Nachfolgekongfigurationen hat, soll der zugehörige Knoten trotzdem r Söhne haben, und die überzähligen Konfigurationen sind leer.

Indeterminierte Turing-Maschinen (NTMs)

Beweis (Fortsetzung)

Darstellung des aktuellen Pfades im Rechnungsbaum als Zahl im r -adischen System.

Eine Zahl $d_1 \dots d_n$ bedeutet:

- Von der Startkonfiguration C_0 aus ist die d_1 -te der r möglichen Nachfolgekongfigurationen gewählt worden, C_{d_1} .
- Von C_{d_1} , einem Knoten der Tiefe 1, aus wurde die d_2 -te mögliche Nachfolgekongfiguration gewählt,
- usw.

Indeterminierte Turing-Maschinen (NTMs)

Beweis (Fortsetzung)

Ausführung des Iterative Deepening:

- Beginne mit 0 auf zweitem Band.
- Jeweils nächste zu betrachtende Rechnung durch Erhöhen der Zahl auf Band 2 um 1
- Auf Band 3 wird eine Rechnung von \mathcal{M} **determiniert** simuliert.
Und zwar entsprechend der Zahl $d_1 \dots d_n$ auf Band 2.
Die Endkonfiguration $C_{d_1 \dots d_n}$ dieser Rechnung steht im Rechenbaum an dem Knoten, der das Ende des Pfades $d_1 \dots d_n$ bildet.
- Ist die Konfiguration $C_{d_1 \dots d_n}$ eine Haltekonfiguration, so hält \mathcal{M}' .
- Sonst Zahl auf Band 2 erhöhen und die nächste Rechnungssimulation beginnen

Indeterminierte Turing-Maschinen (NTMs)

Beweis (Ende)

Damit gilt:

\mathcal{M}' hält bei Input w

gdw

es gibt in R_{C_0} eine Haltekonfiguration.

Das ist genau dann der Fall, wenn

- \mathcal{M} bei Input w hält,
- w in L liegt.

□

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 TMs 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

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

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:

$$\begin{array}{l}
 Z S 2\lambda S 2\lambda \\
 Z S 00 S 2\lambda
 \end{array}$$

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 die es keine TM gibt, die sie aufzählt.

⇒ es gibt Sprachen für die 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 Eingabe 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